

**HYPEREDGE REPLACEMENT:
GRAMMARS AND LANGUAGES**

Vom Fachbereich Mathematik und Informatik
der Universität Bremen
zur Verleihung des akademischen Grades

DOKTOR DER NATURWISSENSCHAFTEN

genehmigte Dissertation

von

Annegret Habel

Bremen 1989

**HYPEREDGE REPLACEMENT:
GRAMMARS AND LANGUAGES**

Vom Fachbereich Mathematik und Informatik
der Universität Bremen
zur Verleihung des akademischen Grades

DOKTOR DER NATURWISSENSCHAFTEN

genehmigte Dissertation

von

Annegret Habel

Bremen 1989

Gutachter:

Prof. Dr. Hans-Jörg Kreowski (Universität Bremen)

Prof. Dr. Hartmut Ehrig (Technische Universität Berlin)

Kolloquium: 18. April 1989

Acknowledgments

The work reported here has been carried out during my affiliation with the Computer Science Departments of the Technical University of Berlin and the University of Bremen.

I would like to express my gratitude to the following people, who made essential contributions to this thesis:

To my thesis supervisors, Hartmut Ehrig (Berlin) and Hans-Jörg Kreowski (Bremen), for their advice, support, and patience. They introduced me to the fascinating world of graph grammars, taught me to do research in theoretical computer science, and gave me an environment in which I enjoyed to do creative work.

To Walter Vogler for many stimulating discussions that we have had during the preparation of joint papers.

To Udo Hummert, Clemens Lautemann, Karin Limberg, Michael Löwe, Detlef Plump, and Anne Wilharm for reading drafts of the thesis, for contributing good ideas, making useful comments and criticisms, for general moral support, and for combinations of the above.

Finally, and most of all, I would like to thank my husband, Christopher, for his support and understanding during the developing stages of this thesis.

Contents

GENERAL INTRODUCTION	1
I. INTRODUCTION TO HYPEREDGE-REPLACEMENT GRAMMARS	4
1. Hyperedges and Hypergraphs	5
2. Replacement of Hyperedges by Hypergraphs	12
3. Hyperedge-Replacement Grammars	16
4. Modifications of HRG's	25
5. Related Concepts	35
6. Bibliographic Note	45
II. BASIC PROPERTIES OF HRG's	46
1. Parallel Derivations	47
2. A Context-Freeness Lemma	49
3. Derivation Trees	54
4. Bibliographic Note	57
III. CHARACTERIZATIONS OF HRL's	58
1. Closure Properties	59
2. A Kleene-Type Characterization	66
3. A Fixed-Point Theorem	69
4. Bibliographic Note	71
IV. STRUCTURAL ASPECTS OF HRL's	72
1. Simplifications of HRG's	73
2. A Pumping Lemma for HRL's	85
3. Parikh's Theorem for HRL's	95
4. Boundedness Properties of HRL's	100
5. Generative Power of HRG's	108
6. Bibliographic Note	127
V. GRAPH-THEORETIC ASPECTS OF HRL's	128
1. Preliminaries	130
2. Some Compatible Graph-Theoretic Properties	132
3. A General View of Compatible Predicates	141
4. Some Compatible Graph-Theoretic Functions	154
5. A General View of Compatible Functions	161
6. Bibliographic Note	167
VI. EXTENSIONS AND VARIATIONS OF HRG's	168
1. Hypergraph-Replacement Grammars	169
2. Hypergraph-Replacement Grammars with Application Conditions	176
3. Parallel Hyperedge-Replacement Grammars	181
4. Figure-Generating Grammars Based on Hyperedge-Replacement	184
5. Bibliographic Note	192
CONCLUSION	193
REFERENCES	194
INDEX	200



Digitized by the Internet Archive
in 2026

General Introduction

Graph grammars and graph languages have been studied since the late sixties and have been motivated by various application areas such as pattern recognition, semantics of programming languages, compiler description, data base systems, specification of data types, developmental biology, etc. (see [CER 79], [ENR 83], and [ENRR 87] for a survey). The general approaches (see, e.g., [Eh 79] and [Na 79]) cover lots of potential applications; the more specific approaches (for the special case of node rewriting, see, e.g., Janssens and Rozenberg [JR 80a+b] and, for the case of edge replacement, see the approach in [Kr 77a+79], [HK 83+85]) promise a rich mathematical outcome.

In this thesis, I propose a kind of compromise by introducing hyperedge-replacement grammars reviving some ideas from the early seventies which can be found in Feder's [Fe 71] and Pavlidis' [Pa 72] work. On the one hand, Petri nets, functional expressions, program flow diagrams, molecule structures, and many other complex information structures can be interpreted in a natural way as hypergraphs (the hyperedges of which may have many "tentacles" to grip at their source and target nodes rather than only two "arms" as the edges of ordinary graphs). On the other hand, the replacement of hyperedges provides a type of context-free rewriting on the level of hypergraphs saving many of the fundamental properties of context-free rewriting on strings and ordinary graphs. Hence, I am confident that this framework combines the theoretical attractivity of a class of context-free grammars and languages with the prospect for nice applications.

In Chapter I, hyperedge-replacement grammars are introduced. They provide a simple mechanism for deriving hypergraphs from hypergraphs by applying productions. In my opinion, hyperedge replacement provides a framework for an attractive hypergraph language theory. Nevertheless, the generative power is large enough to cover hypergraph languages which are interesting from a practical point of view, too.

In Chapter II, basic properties of hyperedge-replacement grammars are considered. In particular, it turns out that hyperedge-replacement works absolutely local without any effect on the context of the hyperedges replaced. Every derivation of a hypergraph H is completely described by corresponding derivations of the hyperedges of H (or, more generally, of an hyperedge-disjoint covering of H).

Some characterizations of hyperedge-replacement languages may illustrate the character of hyperedge replacement. First, Gruska's Kleene-type characterization of context free string languages in [Gr 71] can be carried over to hyperedge-replacement grammars. Secondly, the hypergraph languages generated by hyperedge-replacement grammars turn out to be the smallest fixed points of their generating productions (when they are considered as systems of language equations). Therefore, the construction of smallest fixed points provides an alternative method to generate hypergraph languages. This generalizes Ginsburg's and Rice's well-known Fixed-Point Theorem for context free string languages in [GR 62]. These results, discussed in Chapter III, are evidence that hyperedge replacement grammars represent a hypergraph-grammar version of context-freeness.

In Chapter IV, structural aspects of hyperedge-replacement languages are investigated. In particular, a Pumping Lemma is stated, which generalizes those Pumping Lemmata known for context-free string and graph languages. Roughly speaking, the lemma says that each sufficiently large hypergraph in a hyperedge-replacement language decomposes into $FIRST \otimes LINK \otimes LAST$ such that $FIRST \otimes LINK^i \otimes LAST$, for $i \geq 0$, belongs to the given language. The crucial part is to find the proper composition $\otimes$ for hypergraphs. The Pumping Lemma can be used to derive several boundedness properties of hyperedge-replacement languages (e.g., the boundedness of the connectivity). This leads to the following situation:

On the one hand, it can be shown that many (hyper)graph languages interesting from the point of view of graph theory (for example, series-parallel graphs, outerplanar graphs, k -trees, graphs with bandwidth $\leq k$, graphs with cyclic bandwidth $\leq k$, etc.) can be generated by hyperedge-replacement grammars.

On the other hand, one may apply the Pumping Lemma (and some derived theorems) to various languages for proving that they cannot be generated by hyperedge-replacement grammars. In particular, it turns out that the generative power of hyperedge-replacement grammars depends strongly on the order (this is the upper bound for the number of "tentacles" a non-terminal hyperedge may have for gripping to its source and target nodes). In other words, the order of hyperedge-replacement grammars induces an infinite hierarchy of classes of hypergraph languages, which remains infinite even if one considers graph or string languages, only. The string(-graph) languages include certain non-context-free languages, and the string(-graph) languages of order 2 are just those which can be generated by context-free string grammars.

In Chapter V, some specific graph-theoretic aspects of hyperedge-replacement languages are discussed. Actually, graph-theoretic properties which are "compatible" with the derivation process of hyperedge-replacement grammars are investigated. E.g., connectivity, the existence of Eulerian paths and cycles, edge-colorability, and many other properties can be shown to be of this type. If P is such a compatible property, then for all hyperedge-replacement grammars, the following questions turn out to be decidable.

- (1) Is there a hypergraph in the generated language satisfying the property P ?
- (2) Do all hypergraphs in the generated language satisfy the property P ?

We refer to this result as a metatheorem because each instantiation of the property yields a particular decision result.

Beside compatible properties, "compatible" numeric functions (like the number of nodes and hyperedges, the degree, the number of simple paths, the maximum length of simple paths, the size of a maximum set of independent nodes, etc.) are introduced and a further type of decidability problems, called Boundedness Problems, concerning functions on hypergraphs is investigated. This is as follows:

- (3) Are the values of all hypergraphs in the generated language bounded?

It is shown that such a Boundedness Problem is decidable for a class of hyperedge-replacement grammars if the corresponding quantity function is built up by maxima, sums and products and if the function is compatible with the derivation process of the given grammars.

Finally, in Chapter VI, we discuss extensions of hyperedge-replacement grammars to hypergraph-replacement grammars, to hypergraph grammars with application conditions, and to parallel hyperedge-replacement grammars as well as a variation to so-called "figure-generating" hyperedge-replacement grammars.



I. Introduction to Hyperedge-Replacement Grammars

In many fields of computer science it is suitable to represent information by some kind of diagrams rather than by text; in such cases, information processing concerns the manipulation of high-dimensional structures (like graphs and hypergraphs) rather than linear structures (strings). This is the general motivation to develop a graph-grammar and graph-language theory as a counterpart to formal (string-) language theory. Therefore, the research on graph grammars ranges from various applications in pattern recognition, data base systems, semantics of programming languages etc. to the boundary between graph grammars and string grammars (see [CER 79], [ENR 83], and [ENRR 87] for an overview).

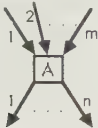
It would be easy to introduce a complicated mechanism of graph rewriting so that the general framework becomes nearly unfeasible, the theoretical achievements are rare and poor, and the best you can achieve are lots of undecidability results. To do somewhat better, many researchers have been looking for simpler special cases of graph rewriting — choosing the atomic items of a graph as key. A majority has proposed various kinds of *node replacements* (see, e.g., Pratt [Pr 71], Della Vigna and Ghezzi [DG 78], Franck [Fr 78], Janssens and Rozenberg [JR 80a+81], Kaul [Ka 83], Lichtblau [Li 85]), where, in particular, Janssens' and Rozenberg's NLC-approach is most successful (see, e.g., Turán [Tu 83], Rozenberg and Welzl [RW 86a+b], Engelfriet and Leih [EL 88]). A minority has been attracted by the (hyper)edges, the alternative atomic items, and proposed (*hyper*)*edge replacement* as a simple way of deriving (hyper)graphs from (hyper)graphs (see, e.g., Feder [Fe 71], Pavlidis [Pa 72], Kreowski [Kr 77a+79], Habel and Kreowski [HK 83+85+87b], Bauderon and Courcelle [BC 87a], Main and Rozenberg [MR 87b], Montanari and Rossi [MR 87]).

In this chapter, I will give a systematic introduction into hyperedge-replacement grammars reviving some of the ideas from the early seventies, e.g., those of Feder [Fe 71] and Pavlidis [Pa 72], and relate them to the concepts in the literature. In Section 1, hypergraphs are introduced as a generalization of ordinary graphs. Their basic units are hyperedges, i.e., objects, which may have many “tentacles” to grip at their source and target nodes rather than only two “arms” as the edges of ordinary graphs. Several high-dimensional structures like Petri nets, functional expressions, program flow diagrams, molecule structures, and many other complex information structures can be represented in a natural way by this kind of hypergraphs. The key construction in this thesis is the replacement of some hyperedges of a hypergraph by a hypergraph. A formal definition of hyperedge replacement is given in Section 2. Based on this construction, in Section 3, so called hyperedge-replacement grammars and languages are introduced. Several examples may illustrate this kind of grammars. These grammars provide a type of context-free rewriting on the level of hypergraphs saving many of the fundamental properties of context-free rewriting on strings and ordinary graphs. Section 4 is concerned with some modifications of hyperedge-replacement grammars (HRG's), which allow to compare this type of grammar with those in the literature. Finally, in Section 5, hyperedge-replacement grammars are related to other graph-grammar formalisms.

1. Hyperedges and Hypergraphs

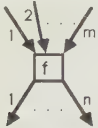
This section provides the basic notions concerning hyperedges and hypergraphs.

In the proposed approach, a hyperedge is an atomic item with an ordered set of incoming tentacles and an ordered set of outgoing tentacles where each tentacle grips at a node through the source and target functions.

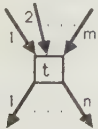


Hyperedges are frequently used in computer science, for instance, as:

1. Functional symbol with arity m and coarity n



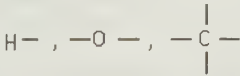
2. Event or transition of a Petri net (with an arbitrary order on the tentacles)



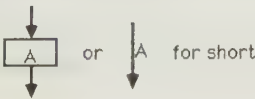
3. Control unit in a flow diagram like statement and decision (with one entry and one exit or two exits, respectively)



4. Atomic chemical unit like hydrogen, oxygen, and carbon (where the order of the only incoming tentacles is clockwise and the names of the atoms are the labels)



5. Ordinary directed edge (with one incoming and one outgoing tentacle)



A set of nodes together with a collection of such hyperedges (usually with varying numbers of tentacles) forms a hypergraph if each tentacle is connected with a node.

1.1 Definition (hypergraphs)

Let C be an arbitrary, but fixed set, called set of *labels* (or *colors*).

1. A (*directed, hyperedge-labeled*) hypergraph over C is a system (V, E, s, t, l) where V is a finite set of *nodes* (or *vertices*), E is a finite set of *hyperedges*, $s : E \rightarrow V^*$ and $t : E \rightarrow V^{* - 1}$ are two mappings assigning a sequence of *sources* $s(e)$ and a sequence of *targets* $t(e)$ to each $e \in E$, and $l : E \rightarrow C$ is a mapping *labeling* each hyperedge.
2. Given a hypergraph (V, E, s, t, l) , a hyperedge $e \in E$ is called an (m, n) -edge for some $m, n \in \mathbb{N}$ if $|s(e)| = m$ and $|t(e)| = n$.² The pair (m, n) is the *type* of e , denoted by $type(e)$.

Remarks

1. Usually, a directed hyperedge may possess a number of source and target nodes. Beside the proposed approach — assigning a sequence of source nodes and a sequence of target nodes to each hyperedge —, one may proceed by assigning a set of source nodes and a set of target nodes to each hyperedge. The consideration of sequences of nodes — instead of sets of nodes — makes it possible to handle a large number of practically relevant examples (e.g., functional expressions, flow diagrams, etc.) in which the order of the tentacles has to be considered.
2. There are no restrictions with respect to the “form” of hyperedges: different tentacles of a hyperedge may grip to the same node. Moreover, there are no restrictions with respect to the labeling of hyperedges; in particular, hyperedges of different type may have the same label.
3. In drawings of hypergraphs, a dot ($\bullet$) represents a node, and a graphical structure of the form



depicts a hyperedge with sources and targets where the label is inscribed in the box, the i -th arrow incoming into the box starts at the i -th source ($i = 1, \dots, m$) and the j -th arrow outgoing from the box reaches the j -th target ($j = 1, \dots, n$). In other words, the graphical representation makes use of the one-to-one correspondence between hypergraphs and bipartite graphs.

4. Beside directed, hyperedge-labeled hypergraphs, we may consider directed hypergraphs in which the nodes as well as the hyperedges are labeled: A (*directed*) *labeled* hypergraph

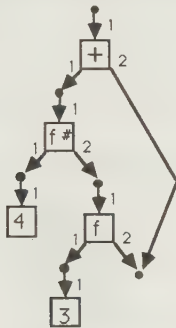
¹ For a set A , A^* denotes the set of all strings over A , including the empty string λ ; $A^+ = A^* - \{\lambda\}$ denotes the set of all strings over A , except the empty string λ .

² For a string $w \in A^*$, $|w|$ denotes the *length* of w .

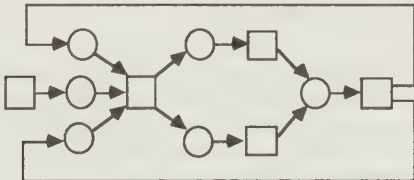
over C is a system (V, E, s, t, l_V, l_E) where (V, E, s, t, l_E) is a hypergraph over C and $l_V : V \rightarrow C$ is a mapping *labeling* each node.

1.2 Examples

1. Hypergraphs built up from functional symbols represent functional expressions like

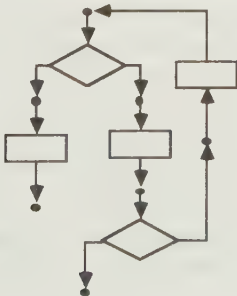


2. A Petri net is usually represented as a directed bipartite graph like

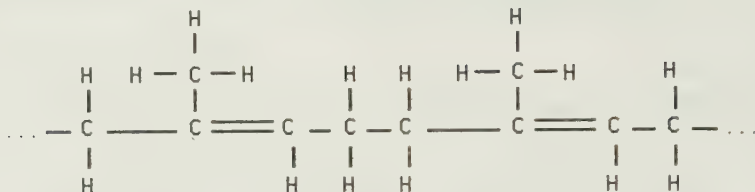


If the transitions (depicted by $\square$) together with their incoming and outgoing edges (ordered arbitrarily) are interpreted as hyperedges, Petri nets, i.e., directed bipartite graphs, turn out to be hypergraphs as defined above.

3. Putting control units together in the usual way, one gets flow diagrams as hypergraphs.



4. A chemical molecule like natural rubber



is a hypergraph if each connection between two atoms is interpreted as a node with two incident tentacles.

5. A undirected graph like the complete bipartite graph $K_{3,4}$



is a hypergraph if each undirected edge abbreviates two directed (1,1)-edges in opposite directions with a special invisible label.

Hyperedges do not only play a static part as building blocks of hypergraphs, but also a more dynamic part as place holders for hypergraphs. However, before a hypergraph can take the place of a hyperedge, it needs some preparation. While a hyperedge grips to its source-nodes and target-nodes according to the conventions, a hypergraph must be equipped with such information. This leads to the concept of a multi-pointed hypergraph which is a hypergraph together with some “external” nodes given by a sequence of begin-nodes (corresponding to the sequence of source-nodes) and a sequence of end-nodes (corresponding to the sequence of target-nodes).

1.3 Definition (multi-pointed hypergraphs)

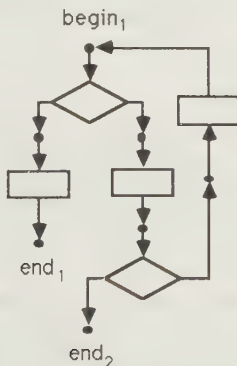
1. A *multi-pointed hypergraph* over C is a system $H = (V, E, s, t, l, \text{begin}, \text{end})$ where (V, E, s, t, l) is a hypergraph over C and $\text{begin}, \text{end} \in V^*$. Components of H are denoted by $V_H, E_H, s_H, t_H, l_H, \text{begin}_H, \text{end}_H$, respectively. The set of all multi-pointed hypergraphs over C is denoted by $\mathcal{H}_C$.
2. $H \in \mathcal{H}_C$ is said to be an (m, n) -hypergraph for some $m, n \in \mathbb{N}$ if $|\text{begin}_H| = m$ and $|\text{end}_H| = n$. The pair (m, n) is the *type* of H , denoted by $\text{type}(H)$.
3. Let H be an (m, n) -hypergraph, $\text{begin}_H = \text{begin}_1 \dots \text{begin}_m$, and $\text{end}_H = \text{end}_1 \dots \text{end}_n$ with $\text{begin}_i, \text{end}_j \in V_H$ (for $i = 1, \dots, m$ and $j = 1, \dots, n$). Then EXT_H denotes the set $\text{EXT}_H = \{\text{begin}_i | i = 1, \dots, m\} \cup \{\text{end}_j | j = 1, \dots, n\}$, called the set of *external nodes* of H . INT_H denotes the set of all other nodes, i.e., $\text{INT}_H = V_H - \text{EXT}_H$, and is called the set of *internal nodes*.

Remarks

1. There is a 1-1-correspondence between hypergraphs and $(0,0)$ -hypergraphs so that hypergraphs may be seen as special cases of multi-pointed hypergraphs. Somewhat ambiguously, we speak about hypergraphs instead of multi-pointed hypergraphs provided that the pointation of the nodes is of a secondary importance.
2. There are no restrictions with respect to the distinguished nodes, i.e., the *begin*- and *end*-nodes of a multi-pointed hypergraph H ; e.g., the sequences $begin_H$ and end_H may have repetitions or a node may be simultaneously considered as the i th and j th *begin* for $i \neq j$.

1.4 Examples

1. A state graph of a finite automaton with its initial and final states provides a nice example of a multi-pointed graph.
2. The external nodes of a flow diagram are its entries and exits so that they correspond to the *begin-end* bracket of a program.



3. The *begin*-component of a functional expression indicates its arity, the *end*-component its coarity.

Graphs and strings may be considered as special cases of (multi-pointed) hypergraphs.

1.5 Definition (graphs and string-graphs)

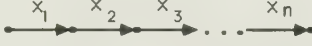
1. A (multi-pointed) hypergraph H over C is said to be a (*multi-pointed*) *graph* if all hyperedges of H are $(1,1)$ -edges.
2. A $(1,1)$ -hypergraph H over C is called a *string graph* if it is of the form $(\{v_0, v_1, \dots, v_n\}, \{e_1, \dots, e_n\}, s, t, l, v_0, v_n)$ where $v_0, v_1, \dots, v_n$ are pairwise different, $s(e_i) = v_{i-1}$, and $t(e_i) = v_i$ for $i = 1, \dots, n$. If $w = l(e_1) \dots l(e_n)$, then the $(1,1)$ -hypergraph over C is called the *string graph* induced by $w \in C^*$ and is denoted by w^* .

Remarks

1. Graphs are drawn as usual, i.e., a (1,1)-edge is drawn as



2. A string graph of the form



provides a unique graph representation of the string $x_1 x_2 x_3 \dots x_n \in C^*$.

1.6 Definition (simple multi-pointed hypergraphs)

1. $H \in \mathcal{H}_C$ is said to be a **frame** if $V_H = EXT_H$ and $E_H = \emptyset$.
2. $H \in \mathcal{H}_C$ is said to be a **singleton** if $V_H = EXT_H$ and $|E_H| = 1$.³
 $e(H)$ refers to the single hyperedge of H and $l(H)$ refers to its label.
3. A singleton H with $s_H(e) = begin_H$ and $t_H(e) = end_H$ is called a **handle**. If $l_H(e) = A$, $type(e) = (m, n)$ for some $m, n \in \mathbb{N}$, and $|EXT_H| = m + n$ (i.e., all nodes of $begin_H$ and end_H are pairwise different), then H is said to be an (m, n) -**handle** induced by A and is denoted by $A(m, n)^*$.

Notations and Remarks

1. For each $H \in \mathcal{H}_C$, $frame(H) = (EXT_H, \emptyset, s, t, l, begin_H, end_H)$ (with the empty mappings s, t , and l) is a **frame**, called the **frame of H** .
2. Given $H \in \mathcal{H}_C$, each hyperedge $e \in E_H$ induces a **handle e^*** by restricting the mappings s_H, t_H , and l_H to the set $\{e\}$, restricting the set of nodes to those occurring in $s_H(e)$ and $t_H(e)$, and choosing $begin_{e^*} = s_H(e)$ and $end_{e^*} = t_H(e)$.

1.7 Definition (subhypergraphs, morphisms, and isomorphic hypergraphs)

1. Let $H, H' \in \mathcal{H}_C$. Then H is called a **subhypergraph** of H' , denoted by $H \subseteq H'$, if $V_H \subseteq V_{H'}$, $E_H \subseteq E_{H'}$, and $s_H(e) = s_{H'}(e)$, $t_H(e) = t_{H'}(e)$, $l_H(e) = l_{H'}(e)$ for all $e \in E_H$. [Note that nothing is assumed on the relation of the distinguished nodes.]
2. Let $H, H' \in \mathcal{H}_C$. A **weak morphism** h from H to H' , denoted by $h : H \rightarrow H'$, consists of a pair of mappings $h = (h_V : V_H \rightarrow V_{H'}, h_E : E_H \rightarrow E_{H'})$ satisfying $h_V^*(s_H(e)) = s_{H'}(h_E(e))$, $h_V^*(t_H(e)) = t_{H'}(h_E(e))$ ⁴, and $l_H(e) = l_{H'}(h_E(e))$ for all $e \in E_H$. A weak morphism $h : H \rightarrow H'$ is said to be a **morphism** from H to H' if $h_V^*(begin_H) = begin_{H'}$ and $h_V^*(end_H) = end_{H'}$.
3. A morphism $h : H \rightarrow H'$ is said to be an **isomorphism** from H to H' if $h_V : V_H \rightarrow V_{H'}$ and $h_E : E_H \rightarrow E_{H'}$ are bijective mappings. $H, H' \in \mathcal{H}_C$ are said to be **isomorphic**, denoted by $H \cong H'$, if there is an isomorphism from H to H' . The class of all hypergraphs isomorphic to a hypergraph H is denoted by $[H]$.

³ For a finite set A , $|A|$ denotes the number of elements in A .

⁴ For a mapping $f : A \rightarrow B$, the free symbolwise extension $f^* : A^* \rightarrow B^*$ is defined by $f^*(a_1 \dots a_k) = f(a_1) \dots f(a_k)$ for all $k \in \mathbb{N}$ and $a_i \in A$ ($i = 1, \dots, k$).

Hypergraphs in the sense of Definition 1.1 (resp. 1.3) are sometimes called *concrete*. In general, the specific sets of nodes and hyperedges chosen to define precisely a hypergraph are irrelevant. Therefore, we shall not distinguish between isomorphic hypergraphs.

1.8 Definition (hypergraph languages)

1. A set $L \subseteq \mathcal{H}_C$ is called a *hypergraph language* over C if it is *closed under isomorphisms*, i.e., if $H \in L$ and $H \cong H'$, then $H' \in L$.
2. A hypergraph language $L \subseteq \mathcal{H}_C$ is said to be *finite* if the number of non-isomorphic hypergraphs in L is finite.

2. Replacement of Hyperedges by Hypergraphs

The key construction in this monograph is the replacement of some hyperedges of a hypergraph by hypergraphs yielding an expanded hypergraph.

An (m, n) -edge can be replaced by a (m, n) -hypergraph in two steps:

- (1) Remove the hyperedge,
- (2) add the hypergraph except the external nodes and hand over each tentacle of a hyperedge (of the replacing hypergraph) which grips to an external node to the corresponding source or target node of the replaced hyperedge.

The construction is rather simple as the following sketch shows.

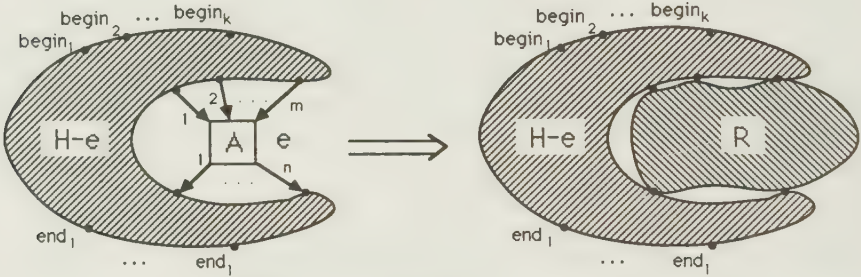


Figure 2.1: Replacement of a Hyperedge.

Beside the replacement of one hyperedge, an arbitrary number of hyperedges may be replaced simultaneously.

2.1 Definition (hyperedge replacement)

1. Let $H \in \mathcal{H}_C$ be a multi-pointed hypergraph, and let $B \subseteq E_H$ be a set of hyperedges to be replaced. Let $repl : B \rightarrow \mathcal{H}_C$ be a mapping with $type(repl(b)) = type(b)$ for all $b \in B$. Then the *replacement* of B in H through $repl$ yields a multi-pointed hypergraph X given by

- $V_X = V_H + \sum_{b \in B} (V_{repl(b)} - EXT_{repl(b)}),^5$
- $E_X = (E_H - B) + \sum_{b \in B} E_{repl(b)},$
- each hyperedge keeps its label,
- each hyperedge of $E_H - B$ keeps its sources and targets,
- each hyperedge of $E_{repl(b)}$ (for all $b \in B$) keeps its internal sources and targets and the external ones are handed over to the corresponding sources and targets of H , i.e.,

⁵ The sum symbols “+” and “ $\sum$ ” denote the disjoint union of sets; the symbol “-” denotes the set-theoretical difference.

$$\left\{ \begin{array}{l} s_X(e) = \text{handover}^*(s_{\text{repl}(b)}(e)) \\ t_X(e) = \text{handover}^*(t_{\text{repl}(b)}(e)) \end{array} \right\} \quad \text{for all } b \in B \text{ and } e \in E_{\text{repl}(b)}$$

where $\text{handover} : V_{\text{repl}(b)} \rightarrow V_X$ is defined by

$\text{handover}(v) = v$ for $v \in V_{\text{repl}(b)} - \text{EXT}_{\text{repl}(b)}$,

$\text{handover}(x_i) = s_i$ ($i = 1, \dots, m$) for $\text{begin}_{\text{repl}(b)} = x_1 \dots x_m$ and $s_H(b) = s_1 \dots s_m$,

$\text{handover}(y_j) = t_j$ ($j = 1, \dots, n$) for $\text{end}_{\text{repl}(b)} = y_1 \dots y_n$ and $t_H(b) = t_1 \dots t_n$,

- $\text{begin}_X = \text{begin}_H$ and $\text{end}_X = \text{end}_H$.

2. The multi-pointed hypergraph X is denoted by $\text{REPLACE}(H, \text{repl})$.

Remarks

1. If $B = \{e\}$ and $\text{repl}(e) = R$, then we write $\text{REPLACE}(H, e \mapsto R)$ instead of $\text{REPLACE}(H, \text{repl})$.

2. The construction above is meaningful and determines (up to isomorphism) a unique hypergraph X if handover is well-defined. This is automatically fulfilled whenever all begin - and end -nodes of each replacing hypergraph are pairwise distinct. If one wants to avoid such a restriction, one has to require that the following condition is satisfied for each $b \in B$: If $\text{begin}_{\text{repl}(b)} = x_1 \dots x_m$ and $\text{end}_{\text{repl}(b)} = x_{m+1} \dots x_{m+n}$ as well as $s_H(b) = y_1 \dots y_m$ and $t_H(b) = y_{m+1} \dots y_{m+n}$, then, for $i, j = 1, \dots, m+n$, $x_i = x_j$ implies $y_i = y_j$.

2.2 Examples

1. If we replace the (1,2)-edge e (with label f) of the hypergraph H in Figure 2.2 by the (1,2)-hypergraph R we obtain the hypergraph X .

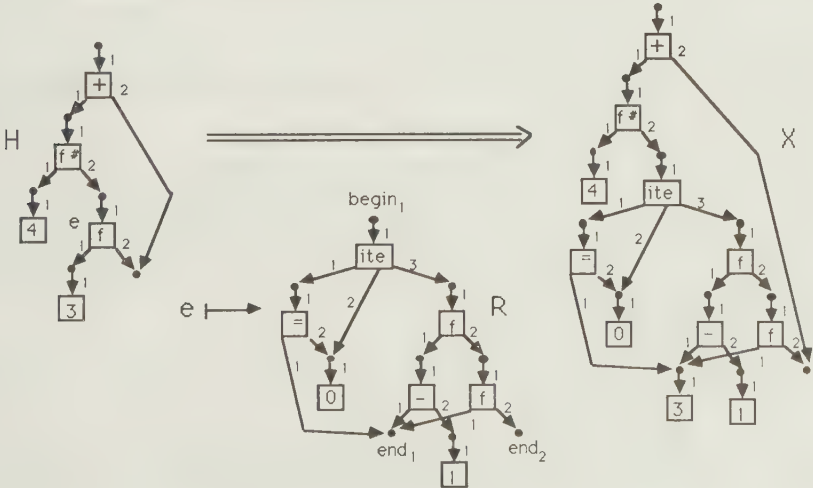
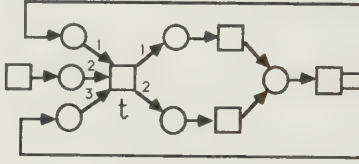
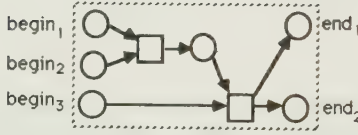


Figure 2.2: Replacement of a Hyperedge.

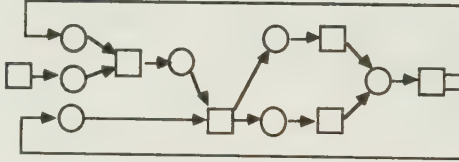
2. Net refinements in the sense of Valette [Va 79] and Suzuki and Murata [SM 80] are hyperedge replacements. For instance, the replacement of the transition t in



by the net



yields the net



The core of the following considerations is the *REPLACE*-operation. Therefore, let us mention some properties of the operation *REPLACE*.

2.3 Lemma (properties of *REPLACE*)

Let $H \in \mathcal{H}_C$, $B \subseteq E_H$, and $repl : B \rightarrow \mathcal{H}_C$ be a mapping such that *REPLACE*($H, repl$) is defined.

1. If, for all $e \in B$, $repl(e) = e^*$, then *REPLACE*($H, repl$) $\cong H$.
2. Let $B_1, B_2 \subseteq B$ such that $B_1 \cup B_2 = B$ and $B_1 \cap B_2 = \emptyset$. Then
 $REPLACE(H, repl) \cong REPLACE(REPLACE(H, repl_1), repl_2)$
 where $repl_1$ and $repl_2$ are the restrictions of $repl$ to B_1 and B_2 , respectively.
3. Let $B_i \subseteq E_H$ and $repl_i : B_i \rightarrow \mathcal{H}_C$ a mapping such that *REPLACE*($H, repl_i$) is defined ($i = 1, 2$). Moreover, let $B_1 \cap B_2 = \emptyset$. Then

$$\begin{aligned} & REPLACE(REPLACE(H, repl_1), repl_2) \\ & \cong REPLACE(H, repl_1 + repl_2) \\ & \cong REPLACE(REPLACE(H, repl_2), repl_1) \end{aligned}$$

where $repl_1 + repl_2 : B_1 + B_2 \rightarrow \mathcal{H}_C$ is given by $repl_1 + repl_2(e) = repl_1(e)$ for $e \in B_1$ and $repl_1 + repl_2(e) = repl_2(e)$ otherwise.

4. Let $B' \subseteq \sum_{e \in B} E_{repl(e)}$, and $repl' : B' \rightarrow \mathcal{H}_C$. Then

$$REPLACE(REPLACE(H, repl), repl') \cong REPLACE(H, repl'')$$

where $repl'' : B \rightarrow \mathcal{H}_C$ is given by $repl''(e) = REPLACE(repl(e), repl'|_{E_{repl(e)} \cap B'})$ for all $e \in B$.

Interpretation

Statement (1) means that the replacement of a hyperedge e by the induced handle $e^\bullet$ yields the original hypergraph, again. Statement (2) says that each parallel replacement of hyperedges (according to *repl*) may be done sequential in the sense that we obtain the same result if first some hyperedges are replaced (according to *repl*) and then the remaining hyperedges are replaced. Statement (3) means that each sequential replacement of sets of hyperedges can be done in parallel provided that the sets of hyperedges in consideration are disjoint. Finally, (4) says that each sequential replacement of sets of hyperedges can be done in one step according to a new replacement function built from the old ones.

Proof

Lemma 2.3 immediately follows from Definition 2.1.

□

3. Hyperedge–Replacement Grammars

In this section, I introduce hyperedge–replacement grammars generalizing context-free graph grammars as described in [Kr 77a+79] and context-free string grammars. They can be seen as hypergraph–manipulating and hypergraph–language generating devices and turn out to be closely related to other concepts in computer science (see Section 5).

Based on hyperedge replacement, one can derive multi-pointed hypergraphs from multi-pointed hypergraphs by applying productions of a simple form.

3.1 Definition (productions and derivations)

1. Let $N \subseteq C$ be a set of *nonterminals*. A *production* over N is an ordered pair $p = (A, R)$ with $A \in N$ and $R \in \mathcal{H}_C$ ¹. A is called the *left-hand side* of p and is denoted by $lhs(p)$, R is called the *right-hand side* and is denoted by $rhs(p)$. The *type* of p , denoted by $type(p)$, is given by the type of R .

2. Let $H, H' \in \mathcal{H}_C$, $p = (A, R)$ be a production, and $e \in E_H$ a hyperedge with $l_H(e) = A$ and $type(e) = type(R)$. Then H *directly derives* H' through p (applied to e) if H' is isomorphic to $REPLACE(H, e \mapsto R)$.

We write $H \xRightarrow[p, e]{\quad} H'$, $H \xRightarrow[p]{\quad} H'$, or $H \xRightarrow[p]{\quad} H'$ provided that $p \in P$.

3. A sequence of direct derivations

$$H_0 \xRightarrow[p_1, e_1]{\quad} H_1 \xRightarrow[p_2, e_2]{\quad} \dots \xRightarrow[p_k, e_k]{\quad} H_k$$

is called a *derivation* (of length k) from H_0 to H_k . Additionally, in the case $H_0 \cong H'_0$, we speak about a *derivation* from H_0 to H'_0 of length 0.

A derivation from H to H' is shortly denoted by $H \xRightarrow[p]{\star} H'$ or $H \xRightarrow[p]{\star} H'$. If the length of the derivation should be stressed, we write $H \xRightarrow[p]{k} H'$ or $H \xRightarrow[p]{k} H'$. The set of all derivations (over a given set of productions) is denoted by DER .

Remark

The application of a production $p = (A, R)$ of type (m, n) to a multi-pointed hypergraph H requires the following two steps only:

- (1) Choose a hyperedge e of type (m, n) with label A .
- (2) Replace the hyperedge e in H by R .

In particular, there is no additional application condition employed.

¹ For simplicity, we assume that, for right-hand sides R with $type(R) = (m, n)$, $|EXT_R| = m + n$.

Using the introduced concepts of productions and derivations, hyperedge-replacement grammars (HRG's) and languages (HRL's) can be introduced in a straightforward way.

3.2 Definition (hyperedge-replacement grammars and languages)

1. A *hyperedge-replacement grammar* is a system $HRG = (N, T, P, Z)$ where $N \subseteq C$ is a set of *nonterminals*, $T \subseteq C$ is a set of *terminals*, P is a finite set of *productions* over N and $Z \in \mathcal{H}_C$ is the *axiom*.
2. The *hypergraph language* $L(HRG)$ *generated by* HRG consists of all terminal labeled hypergraphs which can be derived from Z by applying productions of P :

$$L(HRG) = \{H \in \mathcal{H}_T \mid Z \xrightarrow{*} H\}.$$

3. A hypergraph language $L \subseteq \mathcal{H}_C$ is said to be a *hyperedge-replacement language* if there is a hyperedge-replacement grammar HRG with $L(HRG) = L$.
4. Two hyperedge-replacement grammars HRG and HRG' are said to be *equivalent* if $L(HRG) = L(HRG')$.

Remarks

1. Hyperedge-replacement grammars as introduced above are based on the replacement of hyperedges. It turns out (see Chapter II, Section 2) that hyperedge replacement works absolutely local without any effect on the context of the hyperedges replaced. To emphasize this aspect, hyperedge-replacement grammars are sometimes called context-free hypergraph grammars (see, e.g., [HK 87b]).
2. Even if one wants to generate graph languages rather than hypergraph languages, one may use nonterminal hyperedges because the generative power of hyperedge-replacement grammars increases with the maximum number of tentacles of a hyperedge involved in the replacement (see Chapter IV, Section 5).
3. It is not assumed that N and T are disjoint sets. Hence hyperedges with terminal labels may be rewritten. The terminals only serve as a filtering mechanism to define the generated languages, they do not influence the derivation process at all.
4. The set $L(HRG)$ is closed under isomorphisms. Hence, $L(HRG)$ is a hypergraph language in the sense of Definition 1.8. Moreover, $L(HRG)$ is *homogeneous* in the following sense: for all $H, H' \in L(HRG)$, $\text{frame}(H) \cong \text{frame}(H')$. Therefore, a language which contains hypergraphs with non-isomorphic frames cannot be generated by the grammars introduced above.

3.3 Example (control-flow graphs)

Control-flow graphs of “semi-structured” programs, which are used for data flow analysis by Farrow, Kennedy and Zucconi [FKZ 76], can be generated by the hyperedge-replacement grammar specified as follows.

Consider the following productions, given in a kind of Backus–Naur form:

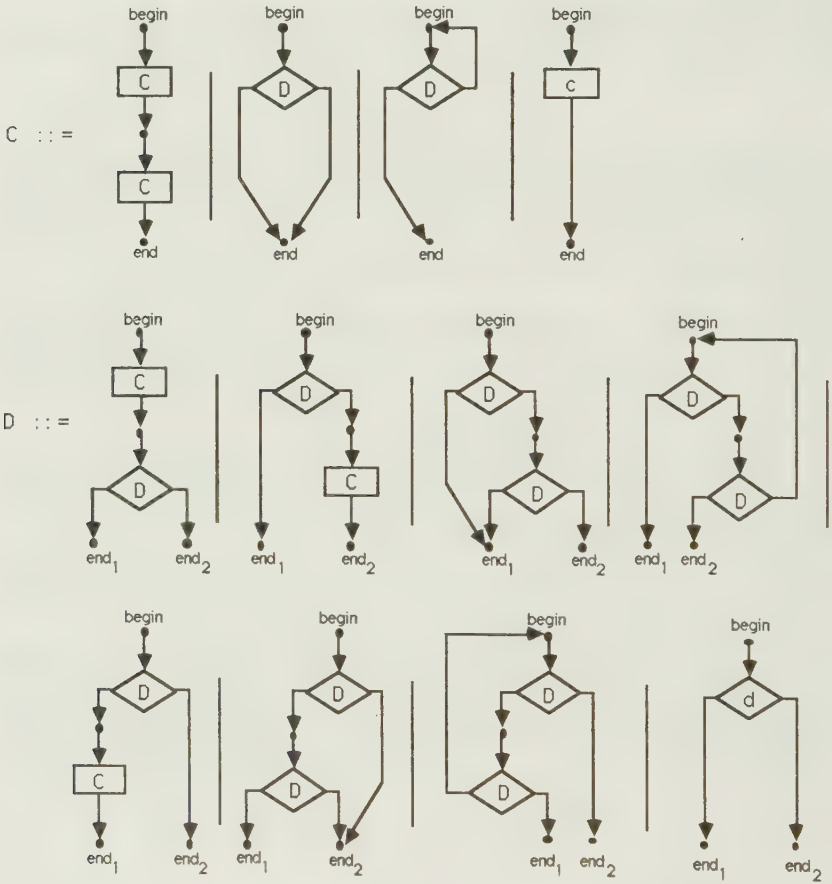


Figure 3.1: Productions for Generating Semi-Structured Control-Flow Graphs.

where the numbering of tentacles is omitted, because it can easily be added. The number of incoming tentacles is always one and the number of outgoing tentacles is one or two.

3.4 Example (chemical structures)

The following hyperedge-replacement grammar generates multi-pointed hypergraphs which describe the chemical structures of the natural rubber molecules. The example stems from Feder [Fe 71], where it is given in terms of so-called plex grammars. Consider the productions

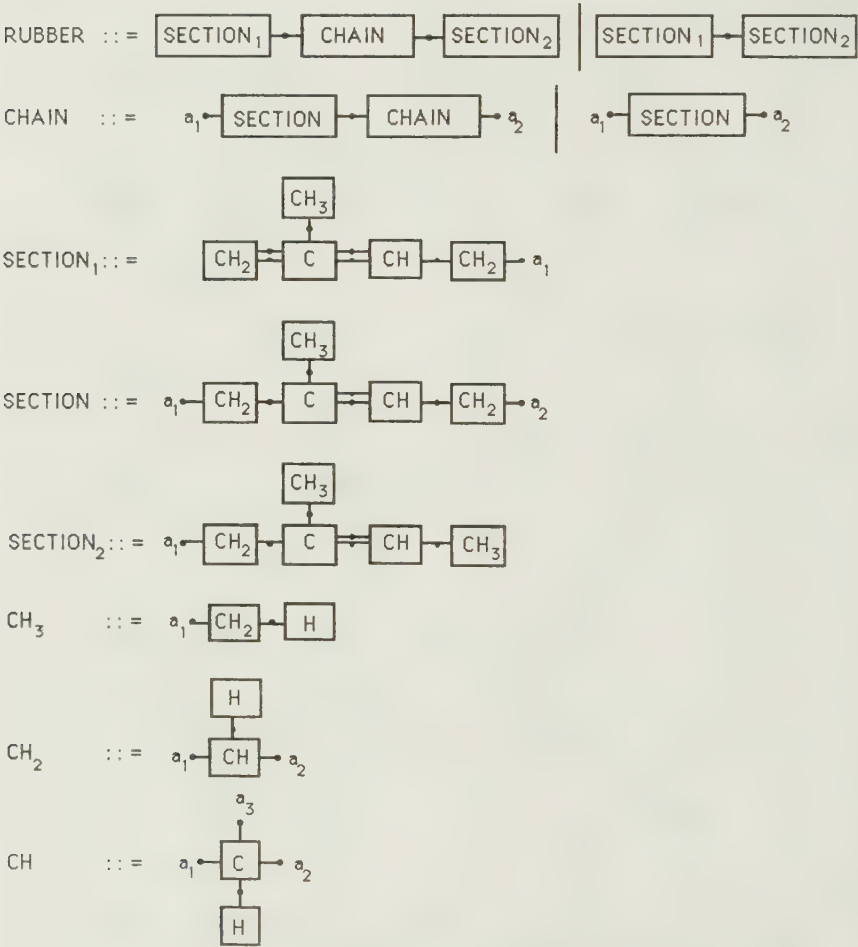


Figure 3.3: Productions for Generating the Chemical Structure of Natural Rubber Molecules.

The numbering as well as the directions of the arrows are omitted, because they are not significant. The attaching points, i.e. the external nodes, are indicated by a_1 , a_2 , and a_3 . Each hyperedge with label *CHAIN* or *SECTION* possesses two attaching points; hyperedges with label *SECTION*₁ or *SECTION*₂ possess one attaching point; and the hyperedges with label *CH*₃, *CH*₂, and *CH* possess one, two, and three attaching points, respectively. Finally, the hyperedges with label *C* and *H* possess four resp. one attaching point.

RUBBER, *CHAIN*, *SECTION*, *SECTION*₁, *SECTION*₂, *CH*₃, *CH*₂, and *CH* are the nonterminal symbols, *C* and *H* (designating the carbon atom and the hydrogen atom) are the terminal symbols. Distinguishing a (0,0)-handle induced by *RUBBER* as axiom, we obtain a hyperedge-replacement grammar, which generates (0,0)-hypergraphs representing the chemical structure of natural rubber molecules.

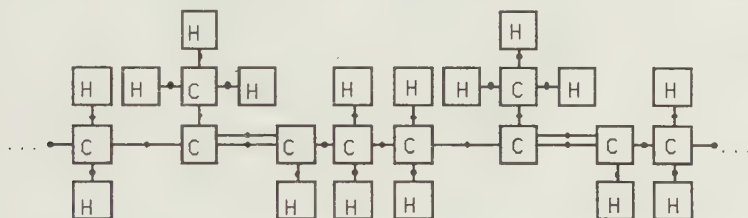


Figure 3.4: Part of a Hypergraph Representing the Chemical Structure of a Natural Rubber Molecule.

3.5 Example (bipartite graphs)

An undirected graph without multiple edges is said to be *bipartite* if its node set V can be partitioned into two non-empty sets V_1 and V_2 such that no pair of two nodes in the same set are adjacent. The set of all bipartite graphs with $|V_1| = k$ is denoted by BIP_k .

The set BIP_k ($k \geq 1$) can be generated by a hyperedge-replacement grammar $HRG_{BIP_k} = (N, T, P, Z)$. The productions of HRG_{BIP_k} are as follows.

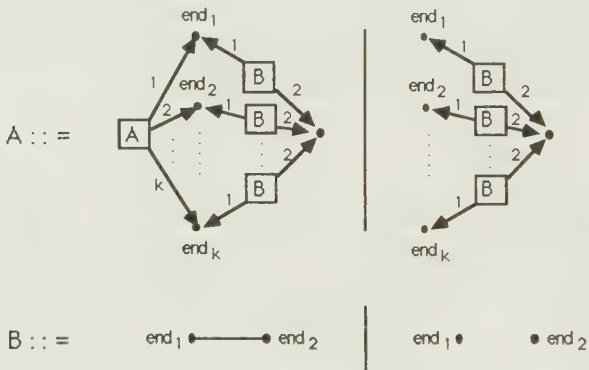


Figure 3.5: Productions of the Grammar HRG_{BIP_k} .

A and B are nonterminals and there is only one terminal label. (Hence terminal hyperedges are sufficiently represented by unlabeled ones.) Finally, a $(0, k)$ -handle induced by the symbol A is used as axiom of HRG_{BIP_k} .

This example demonstrates that hyperedge-replacement grammars can be used to generate sets of graphs. Other graph languages characterized by certain graph-theoretic properties can be generated by hyperedge-replacement grammars, too. Hence the framework may be of some interest even from a graph-theoretic point of view. Further examples like outerplanar graphs, k -trees, graphs of bandwidth $\leq k$, graphs of cyclic bandwidth $\leq k$, etc. are given in Chapter IV, Section 5. A detailed discussion of graph-theoretic aspects of hyperedge-replacement grammars can be found in Chapter V.

3.6 Example (string graphs)

Consider the hyperedge-replacement grammar $HRG = (N, T, P, Z)$ where $N = \{S, A\}$, $T = \{a, b, c\}$, Z is a $(1,1)$ -handle with label S and P consists of the following productions:

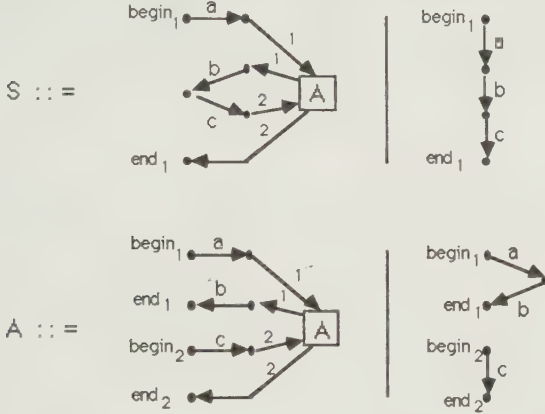


Figure 3.6: Productions of the Grammar HRG .

Starting from the axiom Z , the application of the second production yields the string graph $(abc)^*$. By applying the first production, then applying the third production $n - 1$ times, followed by an application of the fourth production, we obtain the following derivation:

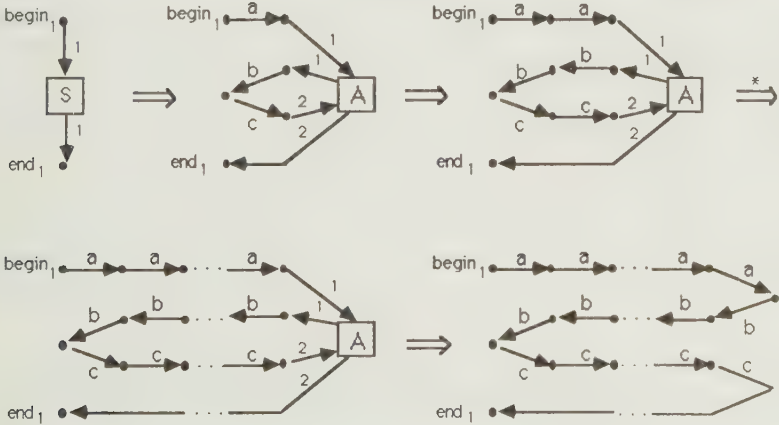


Figure 3.7: Derivation of a String Graph of the Form $(a^{n+1}b^{n+1}c^{n+1})^*$.

Furthermore, the only hypergraphs in $L(HRG)$ are string graphs of the form $(a^n b^n c^n)^*$ for $n \geq 1$. Thus, $L(HRG) = \{(a^n b^n c^n)^* | n \geq 1\}$.

As this example shows, hyperedge-replacement grammars are a powerful tool for generating string-graph languages. In particular, they can be used to generate some string-graph languages whose corresponding string languages are not context-free. Further examples and a detailed discussion can be found in Chapter IV, Section 5.

4. Modifications of HRG's

In Section 3, a hyperedge-replacement grammar $HRG = (N, T, P, Z)$ has been defined in a general manner with respect to different aspects.

- (1) N and T are allowed to be arbitrary sets. In particular, it is not required that the sets are disjoint. Therefore, a symbol may be used as nonterminal symbol as well as as a terminal symbol.
- (2) There are no requirements with respect to the labelings of the hyperedges. In particular, hyperedges of different types may be equally labeled. This gives more flexibility when modelling with this kind of grammars (see, e.g., overloading of operation symbols).
- (3) There are no requirements with respect to the “form” of the hyperedges. In particular, different tentacles of a hyperedge may grip to the same node. This situation occurs, e.g., in the grammar for generating semi-structured control-flow graphs.

In this section, we describe some basic modifications of hyperedge-replacement grammars concerning the nonterminals and terminals, the labelings, and the form of the hyperedges. The Typification Theorem roughly says that each hyperedge-replacement grammar can be transformed into an equivalent one in which same-labeled nonterminal hyperedges are of the same type. The Type-Modification Theorem says that the type of nonterminal hyperedges can be modified in a suitable way. Finally, the Well-Formedness Theorem says that “deformed” hyperedges (i.e., hyperedges having tentacles gripping to the same node) can be replaced by so-called “well-formed” ones.

Comparing hyperedge-replacement grammars $HRG = (N, T, P, Z)$ (as defined in Sect. 3) with usual context-free string grammars, one may observe that — up to now — there are no requirements with respect to the finiteness of N and T , the disjointness of N and T , the labels occurring on the right-hand sides of productions, and the form of the axiom. The decision to define hyperedge-replacement grammars as general as possible is important when modelling application domains with these grammars. In particular, the fact that N and T are not required to be disjoint turns out to be useful. Considering, e.g., the grammar generating chemical structures, one may distinguish the symbols C and H as the only terminal symbols. On the other hand, one may be interested to distinguish, e.g., C and H as well as CH , CH_2 , and CH_3 as terminal symbols. If the set of terminals and the set of nonterminals are allowed to overlap, the set of productions can be retained. Otherwise, we have to modify the set of productions in an unnatural way: it is forced to duplicate some symbols and to consider different versions of a production (where the terminal resp. nonterminal version of a symbol occurs) or to consider additional productions which allow to interpret the symbols as terminal ones.

4.1 Theorem (usual grammars)

For each hyperedge-replacement grammar $HRG = (N, T, P, Z)$, an equivalent grammar $HRG_{\text{usual}} = (N_{\text{usual}}, T_{\text{usual}}, P_{\text{usual}}, Z_{\text{usual}})$ can be constructed such that N_{usual} and T_{usual} are finite, N_{usual} and T_{usual} are disjoint, for $p \in P_{\text{usual}}$, $rhs(p)$ is labeled over $N_{\text{usual}} \cup T_{\text{usual}}$, and Z_{usual} is a singleton the label of which is in N_{usual} .

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. Then an equivalent grammar HRG_{usual} is constructed satisfying the conditions of Theorem 4.1. Firstly, an equivalent grammar with a singleton as axiom is designed. Then, the sets of nonterminals, terminals, and productions are restricted in a suitable way. Finally, some relabelings are done for obtaining a grammar in which nonterminals and terminals are disjoint.

1. Let S_1 be a new symbol not in C and $HRG_1 = (N_1, T, P_1, Z_1)$ where $N_1 = N \cup \{S_1\}$, $P_1 = P \cup \{(S_1, Z)\}$, and Z_1 is a handle with $\text{frame}(Z_1) = \text{frame}(Z)$ and $l(Z_1) = S_1$. Then $L(HRG_1) = L(HRG)$.

2. Let $HRG_2 = (N_2, T_2, P_2, Z_1)$ where $N_2 \subseteq N_1$ and $T_2 \subseteq T$ are the sets of nonterminals resp. terminals occurring in Z_1 or any production and $P_2 \subseteq P_1$ is the set of productions whose right-hand sides are labeled in $N_1 \cup T$. Then N_2 and T_2 are finite because the set P_1 is finite and Z_1 and the right-hand sides of productions are finite. Moreover, the right-hand sides of productions in P_2 are labeled in $N_2 \cup T_2$. Finally, $L(HRG_2) = L(HRG_1)$ because $P_2 \subseteq P_1$ and each production of P_1 which occurs in a derivation from Z_1 to a hypergraph $H \in \mathcal{H}_T$ turns out to be a production in P_2 , too.

3. Finally, we construct a hyperedge-replacement grammar $HRG_3 = (N_3, T_3, P_3, Z_1)$ in which N_3 and T_3 are disjoint. For this purpose, we create a new symbol $\bar{x}$ for each $x \in N_2 \cap T_2$. Let $N_3 = \{x \in N_2 \mid x \notin T_2\} \cup \{\bar{x} \mid x \in N_2 \cap T_2\}$, $T_3 = T_2$, and P_3 be obtained according to the following rule: For each production in P_2 there is a corresponding production in P_3 in which all symbols $x \in N_2 \cap T_2$ are replaced by the corresponding bared ones. Moreover, for each new symbol $\bar{x} \in N_3$, there are productions of the form $(\bar{x}, \hat{x})$ where $\hat{x}$ is a singleton with label x . (Note that the number of this kind of productions can be bounded, because the number of x -labeled hyperedges of different type in P_2 is bounded).

Obviously, N_3 and T_3 are finite and disjoint; moreover, the right-hand sides of productions in P_3 are labeled in $N_3 \cup T_3$, and Z_1 is a singleton whose label does not appear on the right of any production of P_3 . Thus, HRG_3 satisfies the conditions of Theorem 4.1. Moreover, each derivation in HRG_2 can be simulated by a derivation in HRG_3 (choosing the corresponding productions and applying the terminating productions as long as possible). On the other hand, each derivation in HRG_3 has a corresponding derivation in HRG_2 (for each non-terminating production in P_3 , apply the corresponding production in P_2). Therefore, $L(HRG_3) = L(HRG_2)$. $\square$

Notation

Hyperedge-replacement grammars satisfying the conditions of Theorem 4.1 are said to be *usual hyperedge-replacement grammars*.

Considering the definition of hyperedge-replacement grammars given in Section 3, one may wonder why there are no restrictions with respect to the labelings of the hyperedges on the right-hand sides of the productions. This is important with respect to different points. On the one hand, several problems can be easily modeled (e.g., overloading of operation symbols is allowed). On the other hand, when writing productions for a grammar, one

does not have to check syntactic conditions concerning the labeling. Whenever there is a production (A, R) of type (m, n) and a A -labeled hyperedge of different type, then the production cannot be applied to the hyperedge.

With respect to proofs, it is often useful to assume that same-labeled nonterminal hyperedges are of the same type. This situation can be obtained by modifying the grammar in a suitable way. In this case, each occurring nonterminal gets an associated type.

In the following, we consider hyperedge-replacement grammars in which each nonterminal symbol has an associated type. For simplicity, the definition of a “typed grammar” is based on usual hyperedge-replacement grammars.

4.2 Definition (typed grammars)

A usual hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is said to be *typed* if there is a mapping $ltype : N \rightarrow \mathbb{N} \times \mathbb{N}$ such that for each production $(A, R) \in P$

- $ltype(A) = ltype(R)$,
- $ltype(l_R(e)) = ltype(e)$ for all $e \in E_R$ with $l_R(e) \in N$.

4.3 Theorem (Typification Theorem)

For each hyperedge-replacement grammar, an equivalent typed grammar can be constructed.

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. By Theorem 4.1, we may assume that HRG is usual. We construct a grammar $HRG_{type} = (N_{type}, T, P_{type}, Z_{type})$ by “indexing” the labels of nonterminal hyperedges by the types of the hyperedges. (Note that we may change these labels because the nonterminals and the terminals do not overlap.)



Accordingly, we index the symbol on a left-hand side of a production by the type of the hypergraph on the right-hand side.

Formally, $HRG_{type} = (N_{type}, T, P_{type}, Z_{type})$ is given as follows.

- $N_{type} = \{A_{type(R)} \mid (A, R) \in P\} \cup \{l_R(e)_{type(e)} \mid (A, R) \in P, e \in E_R, \text{ and } l_R(e) \in N\}$.
- P_{type} is obtained from P by replacing each production of the form (A, R) by the production $(A_{type(R)}, R_{type})$ where R_{type} is the *typed version* of R , i.e., $R_{type} = (V_R, E_R, s_R, t_R, l, begin_R, end_R)$ where $l(e) = l_R(e)_{type(e)}$ for all $e \in E_R$ with $l_R(e) \in N$ and $l(e) = l_R(e)$ otherwise.
- Z_{type} is the typed version of Z .

Obviously, $HRG_{type} = (N_{type}, T, P_{type}, Z_{type})$ is a usual hyperedge-replacement grammar. Moreover, there is a mapping $ltype : N \rightarrow \mathbb{N} \times \mathbb{N}$ given by $ltype(A_{type(R)}) = ltype(R)$ for

$(A, R) \in P$ and $\text{type}(l_R(e)_{\text{type}(e)}) = \text{type}(e)$ for $(A, R) \in P$, $e \in E_R$, and $l_R(e) \in N$. Thus HRG_{type} is typed. It remains to show that $L(\text{HRG}_{\text{type}}) = L(\text{HRG})$.

To show that $L(\text{HRG}) \subseteq L(\text{HRG}_{\text{type}})$, we use the following observation: Whenever a production (A, R) is used in a derivation $Z \xrightarrow{*} H$, the typed version of the production is used in the derivation $Z_{\text{type}} \xrightarrow{*} H_{\text{type}}$.

Claim 1: Let $Z \xrightarrow{*} H$ be a derivation in HRG . Then there is a derivation $Z_{\text{type}} \xrightarrow{*} H_{\text{type}}$ in HRG_{type} where H_{type} is the typed version of H .

Proof of Claim 1 (by induction on the number of steps in the derivation): The statement is trivial for zero-step derivations: In this case, $Z \cong H$ and $Z_{\text{type}} \cong H_{\text{type}}$. Suppose that it is true for k -step derivations. Let $Z \xrightarrow{*} H$ be a $(k+1)$ -step derivation in HRG . Then the derivation is of the form $Z \xrightarrow[k]{P} G \xrightarrow{(A,R)} H$ where $G \in \mathcal{H}_{N \cup T}$ and $(A, R) \in P$. Let e denote the hyperedge in G with label A replaced in the last step. By inductive hypothesis, there is a derivation $Z_{\text{type}} \xrightarrow{*} G_{\text{type}}$ in HRG_{type} where G_{type} is the typed version of G . In G , (A, R) is applied to e , i.e., $l_G(e) = A$ and $\text{type}(e) = \text{type}(R)$. In G_{type} , e is labeled by $l_G(e)_{\text{type}(e)} = A_{\text{type}(R)}$, i.e., the production $(A_{\text{type}(R)}, R_{\text{type}}) \in P_{\text{type}}$ can be applied to e in G_{type} leading to a multi-pointed hypergraph $\tilde{H}$. It turns out that $\tilde{H} \cong H_{\text{type}}$. Thus, there is a derivation $Z_{\text{type}} \xrightarrow{*} H_{\text{type}}$ in HRG_{type} where H_{type} is the typed version of H . This completes the inductive proof of Claim 1.

By Claim 1, we conclude that $L(\text{HRG}) \subseteq L(\text{HRG}_{\text{type}})$.

To show that $L(\text{HRG}_{\text{type}}) \subseteq L(\text{HRG})$, one simply has to use the following fact: Whenever a production $(A_{\text{type}(R)}, R_{\text{type}}) \in P_{\text{type}}$ is applied to a multi-pointed hypergraph $\tilde{H}$ leading to $\tilde{H}$, the untyped version of the production can be applied to the untyped version of $\tilde{H}$ leading to the untyped version of $\tilde{H}$.

Claim 2: Let $Z_{\text{type}} \xrightarrow{*} \tilde{H}$ be a derivation in HRG_{type} . Then there is a derivation $Z \xrightarrow{*} H$ in HRG where H is the untyped version of $\tilde{H}$, i.e., $H = (V_{\tilde{H}}, E_{\tilde{H}}, s_{\tilde{H}}, t_{\tilde{H}}, l, \text{begin}_{\tilde{H}}, \text{end}_{\tilde{H}})$ where $l(e) = A$ if $e \in E_{\tilde{H}}$ and $l_{\tilde{H}}(e) = A_{\text{type}(e)}$ and $l(e) = l_{\tilde{H}}(e)$ otherwise.

Proof of Claim 2 (by induction on the number of steps): For zero-step derivations, the statement is trivial: In this case, $Z_{\text{type}} \cong \tilde{H}$ and $Z \cong H$ where H is the untyped version of $\tilde{H}$. Suppose that the statement is true for k -step derivations and let $Z_{\text{type}} \xrightarrow{*} \tilde{H}$ be a $(k+1)$ -step derivation in HRG_{type} . Then the considered derivation is of the form $Z_{\text{type}} \xrightarrow[k]{P_{\text{type}}} \tilde{G} \xrightarrow{(\tilde{A}, \tilde{R})} \tilde{H}$ where $(\tilde{A}, \tilde{R}) \in P_{\text{type}}$. By inductive hypothesis, there is a derivation

$Z \xrightarrow{*} G$ in HRG where G is the untyped version of $\tilde{G}$. By definition of HRG_{type} , $(\tilde{A}, \tilde{R})$ is of the form $(A_{\text{type}(R)}, R_{\text{type}})$ where (A, R) is a production in HRG . Let $(\tilde{A}, \tilde{R})$ be applied to the hyperedge $\tilde{e}$ in $\tilde{G}$. Then $l_{\tilde{G}}(\tilde{e}) = A_{\text{type}(R)}$ and $\text{type}(\tilde{e}) = \text{type}(R_{\text{type}}) = \text{type}(R)$. In G (the untyped version of $\tilde{G}$), $l_G(\tilde{e}) = A$ and $\text{type}(\tilde{e}) = \text{type}(R)$. Thus, the production $(A, R) \in P$ can be applied to G leading to a multi-pointed hypergraph H . Obviously, H is isomorphic to the untyped version of $\tilde{H}$. Hence we can conclude that there is a derivation $Z \xrightarrow{*} H$ in HRG where H is the untyped version of $\tilde{H}$. This completes the inductive proof of Claim 2.

Clearly, the untyped version of $\tilde{H} \in \mathcal{H}_T$ equals $\tilde{H}$. Thus $L(HRG_{type}) \subseteq L(HRG)$ by Claim 2, and $L(HRG_{type}) = L(HRG)$. $\square$

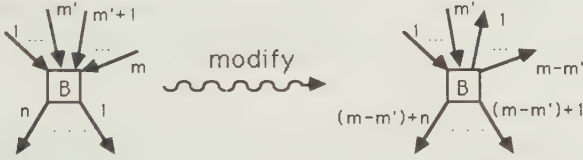
As e.g. the grammar for generating semi-structured control-flow graphs shows, it is adequate to distinguish between incoming tentacles and outgoing tentacles of a hyperedge. With respect to the generated language, it turns out that, for nonterminal hyperedges, it is relevant that there is a tentacle gripping to a node, but not, whether it is an incoming one or an outgoing one.

4.4 Theorem (Type-Modification Theorem)

Let $HRG = (N, T, P, Z)$ be a typed hyperedge-replacement grammar and $B \in N$ a non-terminal of type (m, n) for some $m, n \in \mathbb{N}$. Let $m', n' \in \mathbb{N}$ such that $m' + n' = m + n$. Then there is an equivalent typed hyperedge-replacement grammar $HRG' = (N, T, P', Z')$ in which B is of type (m', n') and all other nonterminals keep their type.

Proof

Let $HRG = (N, T, P, Z)$ be a typed hyperedge-replacement grammar. Then a new grammar $HRG_{mod} = (N, T, P_{mod}, Z_{mod})$ is constructed by replacing the B -labeled hyperedges by B -labeled ones of type (m', n') .



To guarantee that B -labeled hyperedges can be replaced as in the old grammar HRG , for each production $(B, R) \in P$, R must be modified with respect to its type, too.

Formally, HRG_{mod} is constructed according to the following rules:

- P_{mod} is obtained from P by replacing each production (A, R) by the production (A, R') where R' is obtained from R by replacing all B -labeled hyperedges in R by B -labeled hyperedges of type (m', n') and by modifying the type of R (according to the type of A) provided that $A = B$. Formally, $R' = (V_R, E_R, s, t, l_R, \text{begin}, \text{end})$ where $s(e) = \text{first}_{m'}(s_R(e)t_R(e))$ and $t(e) = \text{last}_{n'}(s_R(e)t_R(e))$ ¹ for $e \in E_R$ with $l_R(e) = B$, $s(e) = s_R(e)$ and $t(e) = t_R(e)$ otherwise, $\text{begin} = \text{first}_{m'}(\text{begin}_R \text{end}_R)$ and $\text{end} = \text{last}_{n'}(\text{begin}_R \text{end}_R)$ if $A = B$, and $\text{begin} = \text{begin}_R$ and $\text{end} = \text{end}_R$ otherwise.
- $Z_{mod} = Z$ provided that $l(Z) \neq B$. Otherwise Z_{mod} is obtained from Z by replacing the B -labeled hyperedge in Z by a B -labeled handle of type (m', n') .

¹ For $w \in A^*$, $m, n \in \mathbb{N}$, $\text{first}_m(w)$ denotes the string consisting of the first m symbols of w if $|w| \geq m$, and $\text{first}_m(w) = w$ if $|w| < m$. Analogously, $\text{last}_n(w)$ denotes the last n symbols of w if $|w| \geq n$, and $\text{last}_n(w) = w$ if $|w| < n$. Note that, for $|w| = m + n$, $w = \text{first}_m(w)\text{last}_n(w)$.

Clearly, HRG_{mod} is a typed hyperedge-replacement grammar in which B is of type (m', n') . It remains to show that $L(HRG_{\text{mod}}) = L(HRG)$.

To show that $L(HRG) \subseteq L(HRG_{\text{mod}})$, we use the following fact: Whenever a production $(A, R) \in P$ is applied to a multi-pointed hypergraph G leading to H , the modified production (A, R') can be applied to the modified G leading to the modified H .

Claim: Let $Z \xRightarrow{*} H$ be a derivation in HRG . Then there is a derivation $Z_{\text{mod}} \xRightarrow{*} H_{\text{mod}}$ in HRG_{mod} where H_{mod} is obtained from H by replacing all B -labeled hyperedges — by typedness of HRG , they are of type (m, n) — by B -labeled hyperedges of type (m', n') .

Proof of the Claim (by induction on the number of steps in the derivation): The result is trivial for zero-step derivations: In this case, $Z \cong H$ and $Z_{\text{mod}} \cong H_{\text{mod}}$. Suppose that it is true for k -step derivations. Let $Z \xRightarrow{*} H$ be a $(k+1)$ -step derivation in HRG . Then the derivation is of the form $Z \xRightarrow{*}_P G \xRightarrow{(A, R)} H$ for some $G \in \mathcal{H}_{\text{NUT}}$ and some $(A, R) \in P$. Let

$e \in E_G$ be the hyperedge with label A replaced in the last derivation step. By inductive hypothesis, there is a derivation $Z_{\text{mod}} \xRightarrow{*} G_{\text{mod}}$ in HRG_{mod} where G_{mod} is obtained from G by replacing all B -labeled hyperedges by B -labeled ones of type (m', n') . If $A = B$, then $l_{G_{\text{mod}}}(e) = A$, $\text{type}_{G_{\text{mod}}}(e) = (m', n')$, and $\text{type}(R') = (m', n')$. Otherwise, $l_{G_{\text{mod}}}(e) = A$, $\text{type}_{G_{\text{mod}}}(e) = \text{type}_G(e)$, and $\text{type}(R') = \text{type}(R)$. Hence, $(A, R') \in P_{\text{mod}}$ can be applied to e in G_{mod} leading to a multi-pointed hypergraph $\tilde{H}$. It turns out that $\tilde{H} \cong H_{\text{mod}}$ because G_{mod} is obtained von G by exchange of all B -labeled hyperedges by B -labeled ones of type (m', n') , and R' is obtained from R by exchange of all B -labeled hyperedges by B -labeled ones of type (m', n') . Thus, there is a derivation $Z_{\text{mod}} \xRightarrow{*} H_{\text{mod}}$ in HRG_{mod} . This completes the inductive proof.

Note that $H_{\text{mod}} = H$ for $H \in \mathcal{H}_T$. Now, by the claim, $L(HRG) \subseteq L(HRG_{\text{mod}})$. But it is also true that $L(HRG_{\text{mod}}) \subseteq L(HRG)$, as can be shown in essentially the same manner as it was shown that $L(HRG) \subseteq L(HRG_{\text{mod}})$: Using the same methods, HRG can be “reconstructed” from HRG_{mod} . $\square$

As e.g. the grammar for generating semi-structured control-flow graphs (the second and third production) shows, it is sometimes suitable to distinguish between tentacles of a hyperedge gripping to the same node. With respect to the generated language, it turns out that, for nonterminal hyperedges, it is relevant that there is a tentacle gripping to a node, but not, whether there is more than one tentacle with this property. Although it is often adequate to use “deformed” hyperedges, i.e., hyperedges with tentacles gripping to the same node, a grammar with deformed hyperedges can be modified in such a way that all occurring hyperedges are “well-formed” in the sense that different tentacles of a hyperedge grip to different nodes. (Note that, in the case of the grammar for control-flow graphs, we obtain a grammar which no longer has a natural interpretation.)

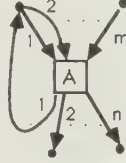
4.5 Definition (well-formed grammars)

1. Let $H \in \mathcal{H}_G$ and $e \in E_H$ with $s_H(e) = x_1 \dots x_m$ and $t_H(e) = x_{m+1} \dots x_{m+n}$ (for some $m, n \in \mathbb{N}$). Then e is said to be *well-formed* if $x_i \neq x_j$ for all $i, j \in \mathbb{N}$ with $1 \leq i < j \leq m+n$.

2. Let $H \in \mathcal{H}_C$ with $\text{begin}_H = x_1 \dots x_m$ and $\text{end}_H = x_{m+1} \dots x_{m+n}$ (for some $m, n \in \mathbb{N}$) and $N \subseteq C$. Then H is said to be *well-formed with respect to N* if all hyperedges $e \in E_H$ with $l_H(e) \in N$ are well-formed and $x_i \neq x_j$ for all $i, j \in \mathbb{N}$ with $1 \leq i < j \leq m+n$.
3. A hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is called *well-formed* if all nonterminal hyperedges in Z are well-formed² and all right-hand sides of productions in P are well-formed (with respect to N).

Remark

A deformed hyperedge (in a hypergraph) looks as follows.



Replacing the deformed hyperedge by a (well-formed) hypergraph R , the hypergraph becomes deformed according to the deformation of the hyperedge. Therefore, the image of R may have deformed hyperedges although the hyperedges in R are well-formed.

4.6 Theorem (Well-Formedness Theorem)

For each hyperedge-replacement grammar, an equivalent well-formed grammar can be constructed.

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. By the Typification Theorem given in 4.3, we may assume that HRG is typed. We now construct a well-formed hyperedge-replacement grammar $HRG_{\text{well}} = (N_{\text{well}}, T, P_{\text{well}}, Z_{\text{well}})$ by “reforming” the right-hand sides of the productions in all possible ways, “redefining” the external nodes, and “replacing” the non-well-formed nonterminal hyperedges by well-formed ones:



[Remark the following: Deformed hyperedges may occur in right-hand sides. Moreover, the application of a production to a deformed hyperedge may yield deformed hyperedges.

² For the axiom Z , it is required that all nonterminal hyperedges are well-formed; according to the specific role of the external nodes of Z ; it is not required that Z itself is well-formed with respect to N . Since the type of Z determines the type of all hypergraphs in the generated language, the type of Z cannot be modified!

Therefore, we create a production for each type of deformation. (This procedure does not change the generated language). Then we exchange deformed hyperedges by well-formed ones: we consider the tentacles of a hyperedge — one after the other — and skip over a tentacle if it grips to the same node as a preceding tentacle. To guarantee that the modified grammar becomes well-formed and generates the same language, we, additionally, modify the external nodes.] More precisely, we are going as follows:

- For each $A \in N$ with $ltype(A) = (m, n)$ (for some $m, n \in \mathbb{N}$) and each equivalence relation $REL \subseteq [m+n] \times [m+n]$, we create a new symbol A_{REL} .³
- Choose $N_{well} = \{A_{REL} | A \in N, ltype(A) = (m, n), \text{ and } REL \subseteq [m+n] \times [m+n] \text{ is an equivalence relation}\}$.
- Construct P_{well} from P according to the following rule: Let (A, R) be a production in P , $begin_R = x_1 \dots x_m$, $end_R = x_{m+1} \dots x_{m+n}$, $REL(R) \subseteq [m+n] \times [m+n]$ be the equivalence relation given by $(i, j) \in REL(R)$ if and only if $x_i = x_j$, and $REL \subseteq [m+n] \times [m+n]$ be an equivalence relation with $REL(R) \subseteq REL$.

Then (A_{REL}, R_{REL}) is a production in P_{well} where R_{REL} is constructed in three steps:

(1) Deformation of R according to REL :

Let $begin_R = x_1 \dots x_m$ and $end_R = x_{m+1} \dots x_{m+n}$. Then REL defines an equivalence relation $\sim_{REL}$ on EXT_R given by $x_i \sim_{REL} x_j$ if and only if $(i, j) \in REL$. Let $\approx_{REL}$ be the extension of $\sim_{REL}$ to V_R and R_1 the quotient hypergraph of R with respect to $\approx_{REL}$.⁴

(2) Redefinition of external nodes:

Let $begin_{R_1} = x_1 \dots x_m$ and $end_{R_1} = x_{m+1} \dots x_{m+n}$. Then R_2 is built from R_1 as $R_2 = (V_{R_2}, E_{R_2}, s_{R_2}, t_{R_2}, l_{R_2}, begin_{R_2}, end_{R_2})$ with $begin_{R_2} = x'_1 \dots x'_m$ and $end_{R_2} = x'_{m+1} \dots x'_{m+n}$ where, for $j = 1, \dots, m+n$, $x'_j = x_j$ if $(i, j) \notin REL$ for all i with $1 \leq i < j$ and $x'_j = \lambda$ (the empty word) otherwise.

(3) Replacement of nonterminal hyperedges by well-formed ones:

R_3 is obtained from R_2 by modification of all nonterminal hyperedges according to the following rule: Let $e \in E_{R_2}$ be a hyperedge with $s_{R_2}(e) = x_1 \dots x_m$, $t_{R_2}(e) = x_{m+1} \dots x_{m+n}$, and $l_{R_2}(e) = A$. Then e defines an equivalence relation $REL(e) \subseteq [m+n] \times [m+n]$ given by $(i, j) \in REL(e)$ if and only if $x_i = x_j$.

Now let $R_3 = (V_{R_3}, E_{R_3}, s_{R_3}, t_{R_3}, l_{R_3}, begin_{R_3}, end_{R_3})$ with $s_{R_3}(e) = x'_1 \dots x'_m$, $t_{R_3}(e) = x'_{m+1} \dots x'_{m+n}$, and $l_{R_3}(e) = A_{REL(e)}$ for nonterminal $e \in E_{R_3}$ (where,

³ For $k \in \mathbb{N}$, $[k]$ denotes the set $\{1, \dots, k\}$.

⁴ Let $R = (V_R, E_R, s_R, t_R, l_R, begin_R, end_R)$ and $\approx$ be an equivalence relation on V_R . Then the quotient hypergraph of R with respect to $\approx$, denoted by $R/\approx$, is the multi-pointed hypergraph X given by

- $V_X = \{[v] | v \in V_R\}$ where the equivalence class $[v]$ is defined by $[v] = \{v' \in V_R | v' \approx v\}$,
- $E_X = E_R$,
- $s_X(e) = [x_1] \dots [x_k]$ and $t_X(e) = [x_{k+1}] \dots [x_{k+l}]$
if $s_R(e) = x_1 \dots x_k$ and $t_R(e) = x_{k+1} \dots x_{k+l}$ for all $e \in E_R$,
- $l_X(e) = l_R(e)$ for all $e \in E_R$,
- $begin_X(e) = [y_1] \dots [y_m]$ and $end_X(e) = [y_{m+1}] \dots [y_{m+n}]$
if $begin_R = y_1 \dots y_m$ and $end_R = y_{m+1} \dots y_{m+n}$.

for $j = 1, \dots, m+n$, $x'_j = x_j$ if $(i, j) \notin REL(e)$ for all i with $1 \leq i < j$ and $x'_j = \lambda$ otherwise) and $s_{R_1}(e) = s_{R_2}(e)$, $t_{R_1}(e) = t_{R_2}(e)$, and $l_{R_1}(e) = l_{R_2}(e)$ for terminal $e \in E_{R_1}$.

Obviously, $R_{REL} = R_3$ is well-formed (with respect to N).

The new set of productions, P_{well} , is finite, because P consists of a finite number of productions and, for each production $(A, R) \in P$ with $ltype(A) = (m, n)$, there is only a finite number of equivalence relations $REL \subseteq [m+n] \times [m+n]$.

- Finally, let Z_{well} be constructed from Z according to the rule given in step (3). Then all hyperedges in Z_{well} are well-formed.

By construction, $HRG_{well} = (N_{well}, T, P_{well}, Z_{well})$ is well-formed. Hence it remains to show that $L(HRG_{well}) = L(HRG)$. To show $L(HRG) \subseteq L(HRG_{well})$, we claim the following:

Claim 1: Let $Z \xRightarrow{*} H$ be a derivation in HRG . Then there is a derivation $Z_{well} \xRightarrow{*} H_{well}$ in HRG_{well} , where H_{well} is obtained from H according to the rule given in step (3).

Proof of Claim 1 (by induction on the number of derivation steps): The statement is trivial for zero-step derivations: In this case $Z \cong H$ and $Z_{well} \cong H_{well}$. Now suppose that it is true for k -step derivations. Consider a $(k+1)$ -step derivation in HRG , say $Z \xRightarrow{k+1} H$. Then the derivation is of the form $Z \xRightarrow{k} G \xRightarrow{(A,R)} H$ for some $G \in \mathcal{H}_{N \cup T}$ and

$(A, R) \in P$. By inductive hypothesis, there is a derivation $Z_{well} \xRightarrow{*} G_{well}$ in HRG_{well} where G_{well} is obtained from G according to step (3). Let e be the hyperedge in G replaced in the last derivation step. Then $s_G(e) = x_1 \dots x_m$, $t_G(e) = x_{m+1} \dots x_{m+n}$ for some $m, n \in \mathbb{N}$, $x_i \in V_G$ ($i = 1, \dots, m+n$), and $l_G(e) = A$. By construction of G_{well} , there is a hyperedge e' in G_{well} with $l_G(e') = A_{REL(e)}$. By construction of HRG_{well} , there is a production $(A_{REL(e)}, R_{REL(e)}) \in P_{well}$. Moreover, $type(R_{REL(e)}) = type_{G_{well}}(e')$. Thus, the production $(A_{REL(e)}, R_{REL(e)})$ can be applied to the well-formed hyperedge e' in G_{well} leading to a multi-pointed hypergraph H' . It turns out that $H' \cong H_{well}$. Hence we can conclude that there is a derivation $Z_{well} \xRightarrow{*} H_{well}$ in HRG_{well} . This completes the inductive proof of Claim 1.

Clearly, $H_{well} = H$ for $H \in \mathcal{H}_T$. Thus $L(HRG) \subseteq L(HRG_{well})$.

To show that $L(HRG_{well}) \subseteq L(HRG)$, one has to use the fact: Whenever a production $(A_{REL(e)}, R_{REL(e)})$ can be applied to a multi-pointed hypergraph $\hat{G} \in \mathcal{H}_{N_{well} \cup T}$ leading to $\hat{H}$, then the production $(A, R) \in P$ can be applied to the underlying deformed version of $\hat{G}$ leading to the underlying deformed version of $\hat{H}$. (Note that the special labeling of the hyperedges allows to reconstruct the deformed version from the well-formed one.)

Claim 2: Let $Z_{well} \xRightarrow{*} \hat{H}$ be a derivation in HRG_{well} . Then there is a derivation $Z \xRightarrow{*} \hat{H}_{deform}$ in HRG where $\hat{H}_{deform}$ is the underlying deformed version of $\hat{H}$, i.e., $\hat{H}_{deform} = V_{\hat{H}}, E_{\hat{H}}, s, t, l, begin_{\hat{H}}, end_{\hat{H}}$ where, for $e \in E_{\hat{H}}$ with $l_{\hat{H}}(e) \in N_{well}$, $s(e) = x_1 \dots x_m$, $t(e) = x_{m+1} \dots x_{m+n}$, and $l(e) = A$ provided that $s_{\hat{H}}(e) = y_1 \dots y_k$, $t_{\hat{H}}(e) = y_{k+1} \dots y_{k+l}$, and $l_{\hat{H}}(e) = A_{REL}$ with $REL \subseteq [m+n] \times [m+n]$, and $x_j = y_{f(j)}$ if $f(j) := \min\{i \mid (i, j) \in REL\}$ and $f(j) < j$ and $x_j = y_{j-k}$ if $f(j) = j$ and $k = |\{i \mid f(i) < i < j\}|$ for $j = 1, \dots, m+n$.

[Note that $(\hat{H}_{deform})_{well} = \hat{H}$ and $(H_{well})_{deform} = H$.]

Proof of Claim 2 (by induction on the number of derivation steps): For zero-step derivations, the statement is trivial: In this case, $Z_{\text{well}} \cong \hat{H}$ and $(Z_{\text{well}})_{\text{deform}} \cong \hat{H}_{\text{deform}}$. Suppose that the statement is true for derivations of length k , and let $Z_{\text{well}} \xrightarrow{k+1} \hat{H}$ be a $(k+1)$ -step derivation in HRG_{well} . Then the derivation is of the form $Z_{\text{well}} \xrightarrow{k} \hat{G} \xrightarrow{(A_{REL}, R_{REL})} \hat{H}$ for

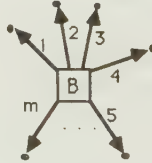
some $\hat{G} \in \mathcal{H}_{N_{\text{well}} \cup T}$ and some $(A_{REL}, R_{REL}) \in P_{\text{well}}$. By inductive hypothesis, there is a derivation $Z_{\text{well}} \xrightarrow{*} \hat{G}_{\text{deform}}$ in HRG where $\hat{G}_{\text{deform}}$ is the underlying deformed version of $\hat{G}$. Let $\hat{e}$ be the hyperedge in $\hat{G}$ replaced in the last derivation step. Then $s_{\hat{G}}(e) = y_1 \dots y_k$, $t_{\hat{G}}(e) = y_{k+1} \dots y_{k+l}$ for some $k, l \in \mathbb{N}$, $y_i \in V_{\hat{G}}$ ($i = 1, \dots, k+l$), and $l_{\hat{G}}(e) = A_{REL}$ with $REL \subseteq [m+n] \times [m+n]$ for some $m, n \in \mathbb{N}$. Moreover, $\text{type}(R_{REL}) = \text{type}_{\hat{G}}(\hat{e})$. By construction of $\hat{G}_{\text{deform}}$, there is a hyperedge e in $\hat{G}_{\text{deform}}$ with $s_{\hat{G}_{\text{deform}}}(e) = x_1 \dots x_m$, $t_{\hat{G}_{\text{deform}}}(e) = x_{m+1} \dots x_{m+n}$, and $l_{\hat{G}_{\text{deform}}}(e) = A$. Moreover, $\text{type}(\hat{R}) = \text{type}_{\hat{G}_{\text{deform}}}(e)$. Thus the production $(A, R) \in P$ can be applied to e in $\hat{G}_{\text{deform}}$ leading to a multi-pointed hypergraph H . It turns out that $H \cong \hat{H}_{\text{deform}}$. Hence there is a derivation $Z \xrightarrow{*} \hat{H}_{\text{deform}}$ in HRG . This completes the inductive proof.

Clearly, $\hat{H}_{\text{deform}} = \hat{H}$ for $H \in \mathcal{H}_T$. Hence we can conclude that $L(HRG_{\text{well}}) \subseteq L(HRG)$. $\square$

At the end of this section, let us use the Type-Modification Theorem and the Well-Formedness Theorem for modifying a grammar into a grammar in which all nonterminal hyperedges are *stars*, i.e., well-formed hyperedges with outgoing tentacles only.

4.7 Corollary (star-replacement grammars)

For each hyperedge-replacement grammar $HRG = (N, T, P, Z)$, there is an equivalent *star-replacement grammar* $HRG_{\text{star}} = (N, T, P_{\text{star}}, Z_{\text{star}})$, i.e., a well-formed and typed hyperedge-replacement grammar in which each nonterminal hyperedge of the grammar is a star.



Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. By Theorem 4.3, we may assume that HRG is typed. Now consider a nonterminal $B \in N$ of type (m, n) where $m, n \in \mathbb{N}$. By the Type-Modification Theorem 4.4, we can construct an equivalent typed grammar in which B is of type $(0, m+n)$. By repeating this process for all nonterminals of the grammar, we get an equivalent typed grammar in which each nonterminal is of type $(0, k)$ for some $k \in \mathbb{N}$. By the Well-Formedness Theorem, the obtained grammar can be modified to a well-formed one. In this grammar, each nonterminal hyperedge is a star. $\square$

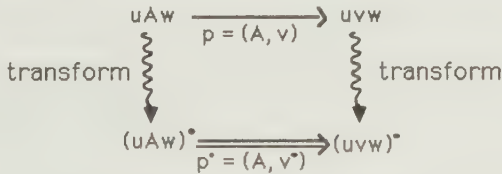
5. Related Concepts

Graph grammars have been developed as an extension of the concept of formal grammars on strings to graphs. Among string grammars, context-free grammars have proven to be extremely useful in practical applications and powerful enough to generate a wide spectrum of interesting formal languages. It is therefore not surprising that analogous notions have been developed also for graph grammars. Corresponding to the different graph-grammar formalisms in the literature, and to the differing opinions about what the term “context-free” means, a number of different types of graph grammars have been given this predicate (cf., Feder [Fe 71], Pavlidis [Pa 72], Della Vigna and Ghezzi [DG 78], Janssens and Rozenberg [JR 80b], Slisenko [Sl 82], Habel and Kreowski [HK 83+85+87b], Montanari and Rossi [MR 87], Bauderon and Courcelle [BC 87], Lengauer and Wanke [LW 88], Lautemann [La 88b]). Among the most general, there are the so-called boundary node-label controlled graph grammars of Rozenberg and Welzl [RW 86a] based on node-replacement and the hyperedge-replacement grammars. In the following, some relationships between the different types of grammars are discussed.

5.1 Context-Free String Grammars

Derivation steps in context-free string grammars can be simulated by direct derivations in hyperedge-replacement grammars using the one-to-one correspondence between strings and string graphs.

Let $p = (A, v)$ be a context-free string production with $A \in N$ ($N \subseteq C$) and $v \in C^+$; let $u, w \in C^*$; and let $uAw \xrightarrow{p} uvw$ denote the resulting derivation step. Furthermore, let $(uAw)^*$, $(uvw)^*$, and v^* be the string graphs corresponding to uAw , uvw , and v , respectively. Then the production $p^* = (A, v^*)$ is applicable to $(uAw)^*$ with the result $(uvw)^*$. This situation is depicted by the following diagram:



Obviously, each context-free string grammar $G = (N, T, P, S)$ (without λ -productions) induces a hyperedge-replacement grammar $G^* = (N, T, P^*, S^*)$ with $P^* = \{p^* | p \in P\}$ such that there is a one-to-one correspondence between the derivations in G and the ones in G^* ; the graph language $L(G^*)$ generated by G^* is obtained from the string language $L(G)$ generated by G just by transformation:

$$L(G^*) = L(G)^* = \{w^* | w \in L(G)\}.$$

In this sense, hyperedge-replacement grammars generalize context-free string grammars. Hence, some results of the string case can be directly carried over to the hypergraph case —

for example all undecidability results known for context-free string grammars. In Chapter IV, Section 5, we will discuss further aspects of the relationship between context-free string grammars and hyperedge-replacement grammars.

5.2 Feder's Plex Grammars

Feder [Fe 71] studies so-called plex grammars. His approach is based on the following idea: In conventional string derivations, each terminal and non-terminal symbol may appear in a string with a symbol to its left or right. Each symbol can be visualized as having two "attaching points", a left one and a right one, at which it may join to or associate with other symbols. A more general symbol can be envisaged as having an arbitrary number, n , of attaching points for joining to other symbols. A symbol of this type is called an n *attaching-point entity*, abbreviated NAPE. Structures formed by interconnecting such entities are called *plex structures*. [The word "plex" is derived from the word "plexus", meaning, "an interwoven arrangement of parts"].

A *Fe-structure* $S = (\Psi, \Gamma, \Delta)$, where the prefix *Fe* refers to Feder, is a system consisting of a *component list* Ψ , a *joint list* Γ , and a *tie-point list* Δ .

- The component list is a string of the form $\Psi = a_1 \dots a_m$ where, for $i = 1, \dots, m$, a_i is a single NAPE, called *component*.
- The joint list Γ specifies the way in which the NAPEs of the component list are interconnected. A joint list is divided into fields. Each *field* is a string of the form $\gamma = \gamma_1 \dots \gamma_m$, where, for $i = 1, \dots, m$, γ_i is a natural number including 0. Each field specifies a *joint*: an entry γ_i in the i th position of a field indicates that attaching point γ_i of the NAPE a_i connects at the joint. If the i th component is not involved at the particular joint, then 0 appears in this position. [Each field must contain at least two non-null identifiers.]
- The tie-point list is also divided into fields. Each *field* is of the form $\delta = \delta_1 \dots \delta_m$, where, for $i = 1, \dots, m$, δ_i is a natural number including 0.

The tie-point fields specify *tie-points* in the same way that the joint-list fields specify joints. [Each tie-point list field must contain at least one non-null identifier.]

To achieve a complete specification, it is required that every attaching point of every NAPE must either connect with another NAPE or be one of the tie-points, i.e., every attaching point of every NAPE must be referenced in at least one field. [Additional requirements are necessary to guarantee that each attaching point of a NAPE is used in at most one field.]

Fe-productions $p = (L, R)$ consist of two *Fe-structures* L and R possessing the same number of tie-points. The correspondence between the tie-points of L and R is given by the ordering of the fields. In particular, a *Fe-production* is said to be *context-free* if the component list of L consists of one component only, the component list of R is non-empty, and R is "connected".

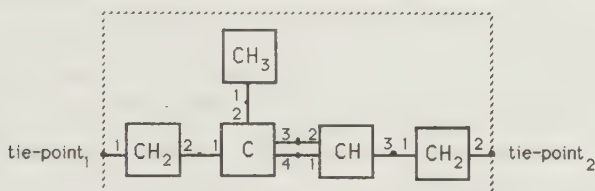
A *Fe-production* $p = (L, R)$ is applied to a connected substructure of a structure G in the following way: Exchange the connected substructure L of G for R .

It turns out that *Fe-structures* and *Fe-productions* correspond directly to multi-pointed hypergraphs and hypergraph productions. Each such direct *Fe-derivation* $G \rightarrow_p H$ corresponds to a direct derivation in the sense of hypergraph replacement if the *Fe-structures* G ,

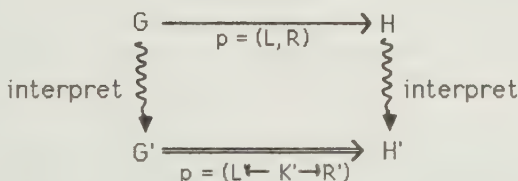
H , L , and R are transformed to multi-pointed hypergraphs by a component-to-hyperedge interpretation, called *interpret*. The operation *interpret* interprets each n attaching-point entity A of the component list of a Fe -structure X by a A -labeled hyperedge with n tentacles and each field of the joint- or tie-point list as a node. Moreover, each field determines all tentacles of hyperedges gripping to the node. Finally, the tie-point list determines some distinguished nodes. For example,

$$S = (\langle CH_2 \rangle \langle C \rangle \langle CH_3 \rangle \langle CH \rangle \langle CH_2 \rangle , \\ (21000, 02100, 03020, 04010, 00031), \\ (10000, 00002))$$

is an Fe -structure. It is shown diagrammed as follows.



The relationship between both kinds of derivations is represented by the following commutative diagram.



where $L' = \text{interpret}(L)$, $R' = \text{interpret}(R)$, and K' is the discrete hypergraph which contains one node for each tie-point of L (resp. R). This leads to a corresponding relation between both kinds of grammars and their generated structures. It turns out that (unrestricted) plex grammars as investigated by Feder can be considered as a special case of hypergraph-replacement grammars in the sense of the Berlin approach (see, e.g., Ehrig [Eh 79] and Chapter VI). Moreover, context-free plex grammars can be considered as hyperedge-replacement grammars.

5.3 Pavlidis' Context-Free Graph Grammars

Pavlidis [Pa 72] studies so-called m th order context-free graph grammars and gives some topological characterizations of sets of graphs which can be generated by this kind of grammars.

An m th order structure is an entity which is connected to the rest of a graphical structure by m nodes. Pavlidis thinks of such elements as polygons which are attached to each other through their nodes. Such structures are closely related to the NAPes proposed by Feder

[Fe 71]. In the following, a structure composed from higher-order structures possessing some distinguished nodes is said to be a *Pa*-structure, where *Pa* refers to Pavlidis. Rewriting such an *m*th order structure corresponds to replacing it by another *Pa*-structure connected to the rest of the structure by exactly *m* nodes.

An *m*th order context-free graph grammar is a system $GG = (N, T, P, Z)$ where

- *N* is a set of nonterminal structures: nodes, branches, triangles, ..., polygons with *m* nodes,
- *T* is a set of terminal structures: nodes and branches,
- *P* is a finite set of productions of the form $p = (A, R)$, where *A* is an *k*th order nonterminal structure (for some $1 \leq k \leq m$) and *R* is a *Pa*-structure (containing possibly both terminal and nonterminal structures) with *k* distinguished nodes,
- *Z* is an initial *Pa*-structure.

Obviously, each *Pa*-structure may be seen as a multi-pointed hypergraph where the nodes of the structures are the nodes of the hypergraph and the structures are resented by hyperedges gripping to the nodes of the structures. It turns out that *m*th order context-free graph grammars as introduced by Pavlidis can be considered as hyperedge-replacement grammars generating graph languages.

5.4 Della Vigna's and Ghezzi's Context-Free Graph Grammars

Della Vigna and Ghezzi [DG 78] study a special kind of node-replacement graph grammars based on the graph-grammar model described by Pratt [Pr 71]. Pratt uses "context-free" graph grammars as the basis grammar from out of which "pair grammars" are constructed¹. Della Vigna and Ghezzi work on the model of Pratt in order to develop a theory on "context-free graph grammars" similar to that of context-free string grammars.

Della Vigna's and Ghezzi's context-free graph grammars work as follows:

DG-graphs $G = (V, E, s, t, l_V, l_E, in, out)$, where the prefix *DG* refers to Della Vigna and Ghezzi, are connected graphs (V, E, s, t, l_E) with an additional node labeling $l_V : V \rightarrow C$ and with two distinguished nodes *in* and *out*.

DG-productions $p = (A, R)$ consist of a nonterminal $A \in N$ and a *DG*-graph *R*.

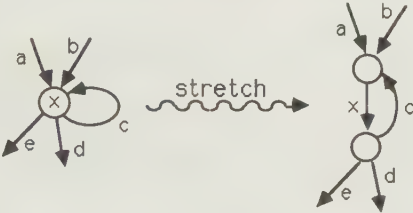
A *DG*-production $p = (A, R)$ is applied to a node *v* of a graph *G* with $l_V(v) = A$ in the following way:

- (1) remove the node *v* from *G*;
- (2) add the right-hand side *R* disjointly;
- (3) edges originally coming into *v* become incoming edges of *in* of *R*;
- (4) edges originally going out from *v* become outgoing edges of *out* of *R*.

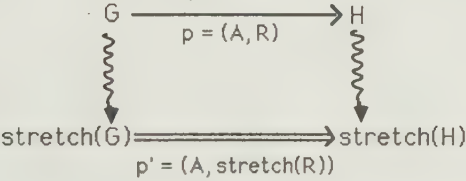
Each such direct *DG*-derivation $G \rightarrow_p H$ corresponds to a direct derivation in the sense of hyperedge replacement if the *DG*-graphs *G*, *H*, and *R* are transformed to (1,1)-graphs by a node-to-edge stretching, called *stretch*. The operation *stretch* replaces each labeled node *v* of a *DG*-graph *X* by two unlabeled nodes *v_s* and *v_t* and a connecting edge *e* such

¹ Pair grammars are used to define translations between string and graph representations of programs and data.

that incoming and outgoing edges as well as labels are inherited according to the following picture. (Note that *in_i* becomes *begin* and *out_i* becomes *end*.)



The relationship between both kinds of derivations is represented by the following commutative diagram.



This leads to a corresponding relation between both kinds of graph grammars and their generated graph languages. It turns out that node-replacement grammars as introduced by Pratt and investigated by Della Vigna and Ghezzi can be considered as a special case of our (hyper)edge-replacement grammars.

This applies in particular to Della Vigna's and Ghezzi's investigations on context-free graph grammars. Hence it is no longer mysterious that the Pumping Lemmata in [DG 78] and [Kr 79] look so similar although they are formulated for graph languages which contain quite different kinds of graphs and are generated by quite different kinds of derivations — at first sight.

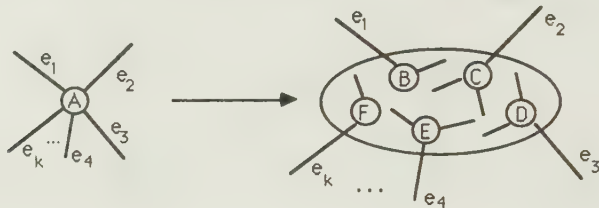
5.5 Slisenko's Context-Free Graph Grammars

Slisenko [Sl 82] studies a special kind of node-replacement graph grammars which work as follows:

Slisenko deals with undirected, node-labeled graphs which are assumed to be connected. *Sli*-graphs $G = (V, E, \varphi, l)$, where the prefix *Sli* refers to the author, are undirected graphs (V, E, φ) with a node labeling $l : V \rightarrow C$. A *star graph* is a triple $R^* = (R, B, b)$, where R is a *Sli*-graph, B is a set of so-called boundary edges (disjoint with the edges of R), and $b : B \rightarrow V_R$ is a mapping assigning a node to each edge in B ². A star graph $R^* = (R, B, b)$ is said to be *simple* if its kernel, R , consists of a single node without edges.

² Each star graph $R^* = (R, B, b)$ may be considered as a partial graph consisting of R and the edges of B where $b : B \rightarrow V_R$ assigns one endpoint to each new edge.

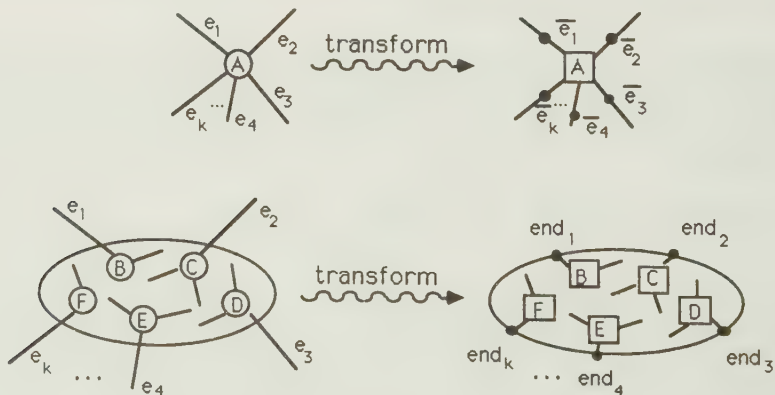
A *Sli*-production $p^* = (L^*, R^*)$ consists of a simple star graph L^* the kernel of which is labeled by a nonterminal $A \in N$ and a star graph R^* with the same boundary set as L^* .



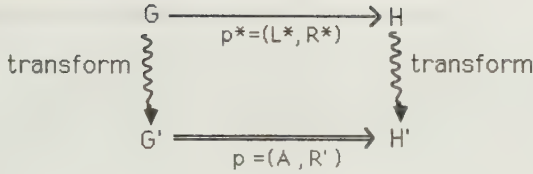
A *Sli*-production $p^* = (L^*, R^*)$ is applied to a node v of a graph G with $l(v) = A$ in the following way:

- (1) remove the node v from G ;
- (2) add the right-hand side R disjointly;
- (3) each edge $e \in B$ originally incident with v becomes incident with $b(e)$ of R .

Each such direct *Sli*-derivation $G \rightarrow_{p^*} H$ corresponds to a direct derivation in the sense of hyperedge replacement if the *Sli*-graphs G, H , and the star graph R are transformed to hypergraphs by a node-to-hyperedge transformation, called *transform*. The operation *transform* replaces each edge e of a *Sli*-graph X by a node $\bar{e}$ and each A -labeled node v of X with k incident edges $e_1, \dots, e_k$ by an A -labeled hyperedge $\bar{v}$ with k tentacles gripping to $\bar{e}_1, \dots, \bar{e}_k$. Moreover, if $e_1, \dots, e_k$ are the edges of the boundary set, then $\bar{e}_1, \dots, \bar{e}_k$ become the distinguished nodes of the hypergraph.



The relationship between both kinds of derivations is represented by the following commutative diagram. (A denotes the label of the simple star graph L^* and $R' = \text{transform}(R^*)$).



This leads to a correspondence between both kinds of graph grammars and their generated graph languages. It turns out that node-replacement grammars as investigated by Slisenko can be considered as a special case of our hyperedge-replacement grammars.

5.6 Montanari's and Rossi's Context-Free Hypergraph Grammars

Montanari and Rossi [MR 87] consider particular classes of networks of constraints called *hierachical networks*. Formally, a network of constrains is seen as a hyperedge-labeled hypergraph whose nodes are variables and whose hyperedges with k tentacles represent k -ary constraints ($k \geq 1$) and thus are labeled by k -ary relations of some finite universe. To describe classes of networks of constraints they use context-free hypergraph grammars. Their grammar model is essentially the same as the one of hyperedge-replacement grammars. The main difference between their definition and the one developed in this thesis is that, for each hyperedge, they do not have functions s and t to identify the source and the target nodes of a hyperedge; they only have one *connection function* c which returns the k -tuple of nodes connected by the hyperedge.

5.7 Bauderon's and Courcelle's Context-Free Graph Grammars

Bauderon and Courcelle [BC 87] define a notion of *context-free graph grammar* based on the substitution of a hypergraph for a hyperedge (of the same type) in a hypergraph. They deal with directed, hyperedge-labeled hypergraphs with a sequence *src* of distinguished nodes called the *sources*. The labels of the hyperedges are chosen in a ranked alphabet, i.e., in a set C each element of which has an associated integer (in $\mathbb{N}$) that they call its *type*. The type is defined by a mapping $ltype : C \rightarrow \mathbb{N}$. The type of the label of a hyperedge must be equal to the number of nodes of that hyperedge.

Each context-free hypergraph grammar in the sense of Bauderon and Courcelle may be seen as a star-replacement grammar in the sense of 4.7 (provided that the sequences *src* do not have repetitions). Moreover, their hypergraphs are more restrictive: terminal as well as nonterminal hyperedges are required to be stars. As the examples in Section 3 have shown, it is adequate to consider also hypergraphs in which the hyperedges are allowed to possess source nodes as well as target nodes:

- For modelling petri nets, it is important to differentiate between the preset and the postset of each transition.
- For representing function expressions, it is adequate to differentiate between the argument list and the value list.
- For representing flow diagrams, it is necessary to differentiate between the *begin* and the *end*.

These possibilities with respect to modelling are not given in the approach of Bauderon and Courcelle.

Finally, let us mention that the approach of Bauderon and Courcelle is mainly oriented by the Berlin Approach to graph grammars (see, e.g., [Eh 79]). Therefore, it is not surprising that hyperedge-replacement grammars in [BC 87] and [HK 87a+b] look so similar.

5.8 Lengauer's and Wanke's Cellular Graph Grammars

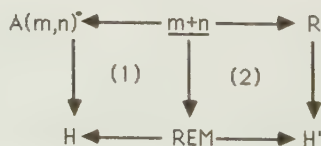
Lengauer and Wanke [LW 88] consider efficient ways of analyzing families of hierarchical engineering designs, using methods from the area of graph grammars. Their work is based on a slightly different version of hyperedge-replacement grammars which they call *context-free cellular graph grammars*.

A *context-free cellular graph grammar* is a list of *cells* $(C_1, \dots, C_k)$ ($k \geq 1$). Each cell C_i is a graph equipped with some hyperedges which are labeled with a set of indices of the index set $\{1, \dots, k\}$. The indices refer to the cells which can be replaced for the hyperedge. [It is required that the hyperedge and the cell corresponding to an index are of the same type.] The generated language of a context-free cellular graph grammar is the set of all cells without hyperedges which can be derived from the cell C_k .

It is easy to see that each context-free cellular graph grammar induces a hyperedge-replacement grammar which generates graphs only.

5.9 Algebraic Approach to Graph Grammars

Let $p = (A, R)$ be a production of type (m, n) , and $H \xRightarrow[p]{\quad} H'$ through $p = (A, R)$ be a direct derivation. Then all the hypergraphs can be grouped together in the way depicted in the following figure, where $\overline{m+n}$ is the discrete hypergraph with m *begin*-nodes and n *end*-nodes (without hyperedges) and REM is obtained from H just by removing the hyperedge which is subject of the replacement. The arrows denote how the hypergraphs are included into each other according to the construction. It is simple to see that squares (1) and (2) form pushout or gluing diagrams so that hyperedge replacement turns out to be a special case of the algebraic approach to (hyper)graph grammars (see, e.g., Ehrig [Eh 79]).



Hence, all results known in the algebraic approach (also called "Berlin approach") apply to hyperedge-replacement grammars.

5.10 Node-Label Controlled Graph Grammars

Janssens and Rozenberg [JR 80a+b] introduce a kind of node-replacement grammars called *node-label controlled* (NLC) grammars. This kind of grammars rewrites single nodes only and establishes connections between the embedded graph and the neighbors of the rewritten node on the basis of the labels of the involved nodes only. This model belongs to one of the most investigated models (see, e.g., Janssens and Rozenberg [JR 81], Brandenburg [Br 83], Turán [Tu 83], Janssens [Ja 83], Janssens, Rozenberg, and Welzl [JRW 86]). Unfortunately, for this kind of grammar, the order of rewriting two nonterminal nodes in a graph may influence the result. Therefore, arbitrary NLC grammars cannot be considered as “truly context-free”. Several ways of enforcing context-freeness in NLC grammars have been investigated so far.

- (1) Janssens and Rozenberg in [JR 80b] restrict NLC grammars to so-called *context-free* NLC grammars. In [Ja 83], this kind of grammars are called *neighborhood-uniform* NLC grammars, short NUNLC grammars, because the embedding is done “uniformly” with respect to neighborhoods within the whole rewriting process. Beside these investigations, in [JKRE 82], so-called *Church-Rosser* NLC grammars are considered. These are NLC grammars with an additional restriction that guarantees that the graph resulting from a derivation is independent of the order in which the direct derivation steps are performed. As shown in [JKRE 82], the context-free NLC grammars studied in [JR 80b] turn out to be special Church-Rosser NLC grammars.
- (2) Rozenberg and Welzl in [RW 86a] restrict NLC grammars to so-called *boundary* NLC grammars, short BNLC grammars. The “boundary” restriction means that all graphs involved (i.e., the axiom and the right-hand sides of productions) are such that two nonterminally labeled nodes are never connected by an edge. Due to the boundary restriction, BNLC grammars are even closer to context-free grammars than arbitrary NLC grammars, and thus have nicer properties with respect to normal forms, decidability, and complexity of recognition (see [RW 86a+b], Welzl [We 87]).
- (3) Recently, Engelfriet and Leih [EL 88] have started to investigate *boundary* eNCE (or B-eNCE) grammars, of which the BNLC grammars are a special case³. These B-eNCE grammars enjoy all the nice properties of the BNLC grammars, whereas they are much easier to handle and understand.
- (4) Finally, let us mention that Courcelle in [Co 88] introduces and investigates so-called *context-free* NLC grammars. Courcelle calls a rewriting system *context-free* if it is *confluent* (two rewriting steps concerning distinct positions can be done in any order, giving the same result) and *associative*. Although NUNLC and BNLC grammars look context-free (they are confluent), they turn out to be not context-free in the sense of Courcelle because they are not necessarily associative.

Recently, some authors have started to investigate the relationships between node-label controlled graph grammars and hyperedge-replacement grammars.

³ eNCE grammars are straightforward generalizations of NCE grammars (see [JR 82]) and can be used to generate sets of graphs which are node as well as edge-labeled. The *e* stands for “edge-labeled” and NCE for “neighborhood controlled embedding”.

(1) Lautemann in [La 88b] shows that each language generated by a BNLC grammar of bounded nonterminal degree can be generated by a hyperedge-replacement grammar, too. Conversely, Vogler in [Vo 88] shows that each set of simple graphs generated by a hyperedge-replacement grammar can be generated by an “enriched” BNLC grammar of bounded nonterminal degree (up to node labels).

(2) Engelfriet and Rozenberg in [ER 88] compare the generating power of the boundary edNCE graph grammars and the hyperedge-replacement grammars. It turns out that boundary edNCE graph grammars of bounded nonterminal degree and hyperedge-replacement grammars have the same power for generating sets of graphs.

5.11 Handle Node-Label Controlled Graph Grammars

While in NLC grammars, the left-hand side of each production is required to be a single node, Main and Rozenberg [MR 87a] examine the extension of NLC grammars where the left-hand side of each production is a *handle*, i.e., a graph with exactly two labeled nodes, and one edge connecting the nodes.

The way that a handle NLC grammar generates a set of graphs is based on the mechanism of ordinary NLC grammars. Roughly speaking, the rewriting process of this kind of grammars works as follows: Given a graph μ and a production $H := \alpha$ (where H is a handle and α is a node-labeled graph) of the grammar, a derived graph is obtained as follows:

- (1) Choose an occurrence of the handle H in μ , called the mother handle. (The set of nodes which are directly connected to the mother handle is called the *neighborhood*.)
- (2) Delete the mother handle (and its incident edges) from the graph μ . Call the resulting graph μ' .
- (3) Add a copy of α (the daughter graph) to the graph μ' .
- (4) Introduce new edges between μ' and the daughter graph according to the following rule: For each pair (Y, Z) in the connection relation C of the grammar, connect every Y -labeled node in the daughter graph with every Z -labeled node in the neighborhood of the mother handle.

Although in edge-replacement grammars as well as in handle NLC grammars an edge (respectively an edge together with its endpoints) is replaced, there are main differences.

- (1) Handle NLC grammars are based on node-labeled graphs while edge-replacement grammars are based on edge-labeled graphs.
- (2) In handle NLC grammars, the handle as well as its incident edges are deleted. Only the nodes in the neighborhood of the handle remain preserved. In edge-replacement grammars, only the edge is deleted; the endpoints of the edge remain preserved.
- (3) Handle NLC grammars have the power of recursive enumerability; edge-replacement grammars do not yield this, because they use a very restrictive mechanism to embed a graph.
- (4) Handle NLC grammars do not have the finite Church-Rosser Property; edge-replacement grammars have this property, an essential property with respect to context-freeness.

5.12 Edge-Label Controlled Graph Grammars

Main and Rozenberg ([MR 87b]) introduce a graph grammar model based on edge replacement, where both the rewriting and the embedding mechanisms are controlled by edge labels. The model of edge-label controlled (ELC) graph grammars is influenced by the model of node-label controlled (NLC) graph grammars as well as by the research on edge-replacement grammars (see [HK 83+85]).

Roughly speaking, the rewriting process of this kind of grammars works as follows: Given a graph μ and a rule $A := (\alpha, \alpha_{source}, \alpha_{target})$ of the grammar, a derived graph is obtained as follows:

- (1) Choose an edge e (mother edge) with label A .
- (2) Delete the mother edge e as well as its source, its target, and its neighborhood edges yielding a new graph μ' .
- (3) Add a copy of α (the daughter graph) to the graph μ' .
- (4) Introduce new edges between μ' and the daughter graph according to the labels of the original neighborhood edges, the labels of edges in the daughter graph which are neighborhood edges with respect to nodes of α_{source} and α_{target} , and the connection relation of the grammar.
- (5) Establish some new internal edges in the daughter graph (provided that originally there were edges parallel to the mother edge).

It is not surprising that the considered model is “too powerful” in the sense that it generates the class of all recursively enumerable graph languages. But, by suitable restrictions, one obtains a graph-grammar model that is — more or less ⁴ — that of edge-replacement grammars.

6. Bibliographic Note

Hyperedge-replacement grammars were first introduced in [HK 87b]. The kind of introduction chosen in Sections 1 to 3 is similar to the one in [HK 87a].

⁴ Edge-replacement grammars as considered in [HK 83+85] allow loops, i.e., edges from a node to itself. There are several sensible ways to handle loops with ELC grammars, but none of these methods corresponds to the treatment given in [HK 83+85].

II. Basic Properties of HRG's

As shown in the previous chapter, several graph-grammar approaches equipped with the predicate "context-free" may be seen as special hyperedge-replacement grammars. In this chapter, we will present some properties of hyperedge-replacement grammars which provide evidence that hyperedge-replacement grammars represent a graph-grammar version of context-freeness.

In Section 1, we generalize the notion of derivations to that of parallel derivations which allows the replacement of an arbitrary collection of hyperedges instead of a single hyperedge. It turns out that parallel derivations can be sequentialized yielding the same hypergraph. In so far, they do not increase the generative power.

In Section 2, we show that hyperedge replacement works absolutely local without any effect on the context of the hyperedges replaced. On the one hand, given a derivation of the form $H \xRightarrow{*} H'$ and a subhypergraph G of H , we are able to extract the part of the derivation concerning G yielding a subderivation $G \xRightarrow{*} G'$ where G' is a subhypergraph of H' . On the other hand, we may consider the subderivations induced by the handles of the hyperedges in H , called *fibres* of the derivation, and combine them to a derivation of H . It turns out that this procedure yields the derivation $H \xRightarrow{*} H'$, again. Therefore, each derivation is uniquely determined by the fibres of the initial hyperedges and the context of these hyperedges. If the replacement of a hyperedge would influence the context, we could not get this result in general.

Finally, in Section 3, we present a notion of a derivation tree for hyperedge-replacement grammars. These derivation trees are quite similar to derivation trees for context-free string grammars which are an essential element of the theory of context-free string grammars; they represent derivations in a hyperedge-replacement grammar. The main problem is given by the graphical structure of a hypergraph. While in a string graph there is an order on the (hyper)edges, we do not have such an order when considering hypergraphs.

1. Parallel Derivations

For technical reasons we are going to generalize the notion of derivations to that of parallel derivations which allows the simultaneous replacement of an arbitrary collection of hyperedges instead of a single hyperedge. We use the notion of a *production base* to determine the choice of hyperedges to be replaced as well as the assignment of productions to be applied.

1.1 Definition (parallel derivations)

1. Let $H \in \mathcal{H}_C$, $B \subseteq E_H$, and P be a set of productions. A mapping $prod : B \rightarrow P$ is called a *production base* in H if $l_H(b) = lhs(prod(b))$ and $type(b) = type(prod(b))$ for all $b \in B$.
2. Let $H, H' \in \mathcal{H}_C$ and $prod : B \rightarrow P$ be a production base in H . Then H *directly derives* H' in *parallel* (through $prod$) if H' is isomorphic to $REPLACE(H, repl)$ where $repl : B \rightarrow \mathcal{H}_C$ is given by $repl(b) = rhs(prod(b))$ for all $b \in B$. In this case, we write $H \xRightarrow[prod]{\Rightarrow} H'$ or $H \xRightarrow[prod]{\Rightarrow} H'$.
3. A sequence of direct derivations

$$H_0 \xRightarrow[prod_1]{\Rightarrow} H_1 \xRightarrow[prod_2]{\Rightarrow} \dots \xRightarrow[prod_k]{\Rightarrow} H_k$$

is called a *parallel derivation* (of length k) from H_0 to H_k . Additionally, in the case $H_0 \cong H'_0$, we speak about a *parallel derivation* from H_0 to H'_0 of length 0.

A parallel derivation from H to H' is shortly denoted by $H \xRightarrow[P]{\Rightarrow} H'$ or $H \xRightarrow[P]{\Rightarrow} H'$. If the length of the derivation should be stressed, we write $H \xRightarrow[P]{k} H'$ or $H \xRightarrow[P]{k} H'$.

4. A direct derivation through $prod : \emptyset \rightarrow P$ is called a *dummy*¹. A parallel derivation is said to be *valid* if at least one of its steps is not a dummy.

Remarks

1. Parallel derivations have some significant properties: On the one hand, the definition of a direct parallel derivation includes the case that all hyperedges are replaced in one step. On the other hand, the definition includes the case that no hyperedge is replaced. Beside these two extrema, a non-empty proper subset of hyperedges may be replaced.
2. A parallel derivation from H to H' through $prod$ is depicted in the following figure.

¹ A production base $prod : B \rightarrow P$ may be *empty*, i.e., $B = \emptyset$. In this case, $H \xRightarrow[prod]{\Rightarrow} H'$ through $prod$ implies $H \cong H'$. In particular, there is always a trivial direct parallel derivation $H \xRightarrow[prod]{\Rightarrow} H$ through $prod$.

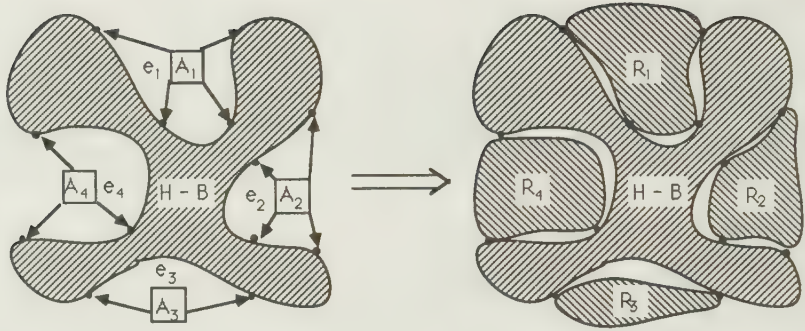


Figure 1.1: Parallel Derivation from H to H' through $\text{prod} : \{e_1, e_2, e_3, e_4\} \rightarrow P$ with $\text{prod}(e_i) = (A_i, R_i)$ ($i = 1, \dots, 4$).

3. Parallel derivations generalize the usual “sequential” derivations in the following sense: For each direct derivation $H \Rightarrow H'$ through $p \in P$ applied to e , there is a direct parallel derivation $H \Rightarrow H'$ through $\text{prod} : \{e\} \rightarrow P$ with $\text{prod}(e) = p$.

1.2 Sequentialization Theorem

Let $H \xRightarrow[\text{prod}]{} H'$ be a direct parallel derivation through $\text{prod} : B \rightarrow P$. Then, for each enumeration $e_1, e_2, \dots, e_n$ of the elements in B , there is a derivation

$$H = H_0 \xRightarrow[\text{prod}(e_1), e_1]{} H_1 \xRightarrow[\text{prod}(e_2), e_2]{} \dots \xRightarrow[\text{prod}(e_n), e_n]{} H_n \cong H'.$$

Proof

The Sequentialization Theorem 1.2 follows immediately from the properties of *REPLACE* presented in I.2.3. $\square$

Remark

The derivations resulting in the Sequentialization Theorem are called *sequentializations* of the given parallel derivation.

1.3 Corollary

$$H \xRightarrow[\text{prod}]{} H' \text{ implies } H \xRightarrow[\text{prod}]{} H'.$$

Proof

Corollary 1.3 is an immediate consequence of the Sequentialization Theorem 1.2. $\square$

Remark

From the viewpoint of generative power, “ $\xRightarrow[\text{prod}]{}^*$ ” and “ $\xRightarrow[\text{prod}]{}^*$ ” have not to be distinguished according to Corollary 1.3 and Remark 1.1(2). Hence, from now on, we use the symbol “ $\xRightarrow[\text{prod}]{}^*$ ” for all kinds of derivations over P .

2. A Context–Freeness Lemma

Most of the results presented in this thesis (as, e.g., the Fixed-Point Theorem and the Pumping Lemma given in Chapter III and IV, respectively) are mainly based on some fundamental aspects of hyperedge-replacement derivations. Roughly speaking, hyperedge-replacement derivations cannot interfere with each other as long as they handle different hyperedges. This is shown in three steps. The Restriction Lemma provides a procedure for restricting a derivation to a subhypergraph. The Joint-Embedding Lemma states that a collection of derivations of the form $H(e) \xRightarrow{*} H(e)'$ can be simultaneously applied to H leading to a single derivation $H \xRightarrow{*} H'$ provided that the $H(e)$ “fit” to hyperedges e in H . Finally, it is shown that first restricting a derivation to each subhypergraph induced by a hyperedge, and then joining them again returns the original derivation. In other words, hyperedge-replacement derivations can be distributed to the handles of the hyperedges without losing information.

2.1 Assumption

For this section, let P be a fixed set of productions.

2.2 Restriction Lemma

Let

$$H_0 \xRightarrow{\text{prod}_1} H_1 \xRightarrow{\text{prod}_2} \dots \xRightarrow{\text{prod}_k} H_k$$

be a derivation where, for $i = 1, \dots, k$, $\text{prod}_i : B_i \rightarrow P$ is a production base in H_{i-1} . Moreover, let $H'_0 \subseteq H_0$. Then there is a derivation

$$H'_0 \xRightarrow{\text{prod}'_1} H'_1 \xRightarrow{\text{prod}'_2} \dots \xRightarrow{\text{prod}'_k} H'_k$$

where, for $i = 1, \dots, k$, $H'_i \subseteq H_i$ and prod'_i is the restriction of $\text{prod}_i : B_i \rightarrow P$ to $B'_i = B_i \cap E_{H'_{i-1}}$.

Notation

The resulting derivation is denoted by $\text{RESTRICT}(\text{der}, H'_0)$ where der refers to the given derivation.

Remarks

1. The explicit construction works as follows: Let $H_0 \xRightarrow{*} H_1$ through $\text{prod}_1 : B_1 \rightarrow P$ be the first derivation step. Consider $B'_1 = B_1 \cap E_{H'_0}$ and the restriction $\text{prod}'_1 : B'_1 \rightarrow P$ of prod_1 to B'_1 . Then prod'_1 is a production base in H'_0 and $H'_0 \xRightarrow{*} H'_1$ can be constructed such that $H'_1 \subseteq H_1$. Hence this restriction step can be iterated.
2. A restriction of a derivation may lead to a derivation with dummy steps.
3. There is a slightly generalized version of the Restriction Lemma saying that whenever $E_{H'_0} \subseteq E_{H_0}$, there is a derivation $H'_0 \xRightarrow{\text{prod}'_1} H'_1 \xRightarrow{\text{prod}'_2} \dots \xRightarrow{\text{prod}'_k} H'_k$ where, for $i = 1, \dots, k$, prod'_i is the restriction of $\text{prod}_i : B_i \rightarrow P$ to $B'_i = B_i \cap E_{H'_{i-1}}$ and $E_{H'_i} \subseteq E_{H_i}$.
4. Let der be a derivation starting from H and $B \subseteq E_H$. Then there is a mapping $\text{fibre} : B \rightarrow \text{DER}$ (induced by der) given by $\text{fibre}(e) = \text{RESTRICT}(\text{der}, e^*)$ for all $e \in B$.

Proof

The Restriction Lemma is proved by induction on the length of the derivation. The basis, $k = 0$, obviously holds. Assume now that the statement holds for some $k \geq 0$. Let $H_0 \xRightarrow{\text{prod}_1} \dots \xRightarrow{\text{prod}_k} H_k \xRightarrow{\text{prod}_{k+1}} H_{k+1}$ be a derivation of length $k + 1$. Then, we may cut off the last derivation step and obtain a derivation of length k . By inductive hypothesis, there is a derivation $H_0 \xRightarrow{\text{prod}'_1} \dots \xRightarrow{\text{prod}'_k} H'_k$ where, for $i = 1, \dots, k$, prod'_i is the restriction of $\text{prod}_i : B_i \rightarrow P$ to $B'_i = B_i \cap E_{H'_{i-1}}$ and $H'_i \subseteq H_i$. In particular, $H'_k \subseteq H_k$ and, hence, $E_{H'_k} \subseteq E_{H_k}$. On the other hand, $B_{k+1} \subseteq E_{H_k}$. Let now $B'_{k+1} = B_{k+1} \cap E_{H'_k}$ and prod'_{k+1} be the restriction of $\text{prod}_{k+1} : B_{k+1} \rightarrow P$ to B'_{k+1} . Since prod_{k+1} is a production base in H_k and $B'_{k+1} \subseteq E_{H'_k} \subseteq E_{H_k}$, prod'_{k+1} is a production base in H'_k . Thus, there is a direct derivation $H'_k \xRightarrow{\text{prod}'_{k+1}} H'_{k+1}$ through prod'_{k+1} where $H'_{k+1} = \text{REPLACE}(H'_k, \text{repl}'_{k+1})$ with $\text{repl}'_{k+1}(e) = \text{rhs}(\text{prod}'_{k+1}(e))$ for $e \in B'_{k+1}$. Hence, we get

$$\begin{aligned} V_{H'_{k+1}} &= V_{H'_k} + \sum_{e \in B'_{k+1}} \text{INT}_{\text{rhs}(\text{prod}'_{k+1}(e))} \\ &\subseteq V_{H_k} + \sum_{e \in B_{k+1}} \text{INT}_{\text{rhs}(\text{prod}_{k+1}(e))} = V_{H_{k+1}}, \\ E_{H'_{k+1}} &= E_{H'_k} - B'_{k+1} + \sum_{e \in B'_{k+1}} E_{\text{rhs}(\text{prod}'_{k+1}(e))} \\ &\subseteq E_{H_k} - B_{k+1} + \sum_{e \in B_{k+1}} E_{\text{rhs}(\text{prod}_{k+1}(e))} = E_{H_{k+1}}. \end{aligned}$$

[Note that $(E_{H_k} - B_{k+1}) \supseteq (E_{H_k} - B_{k+1}) \cap E_{H'_k} = E_{H'_k} - (B_{k+1} \cap E_{H'_k}) = E_{H'_k} - B'_{k+1}$.] Hence, $H'_{k+1} \subseteq H_{k+1}$ and the derivation $H_0 \xRightarrow{\text{prod}'_1} \dots \xRightarrow{\text{prod}'_k} H'_k \xRightarrow{\text{prod}'_{k+1}} H'_{k+1}$ possesses the desired properties. This completes the inductive step. $\square$

2.3 Joint-Embedding Lemma

Let $H \in \mathcal{H}_C$, $B \subseteq E_H$, and $\text{fibre} : B \rightarrow \text{DER}^1$ be a mapping where $\text{fibre}(e)$ for each $e \in B$ is a derivation of the form

$$H(e)_0 \xRightarrow{\text{prod}(e)_1} H(e)_1 \xRightarrow{\text{prod}(e)_2} \dots \xRightarrow{\text{prod}(e)_k} H(e)_k$$

with $H(e)_0 = e^*$ and $k \geq 0$. Then there is a derivation

$$H_0 \xRightarrow{\text{prod}_1} H_1 \xRightarrow{\text{prod}_2} \dots \xRightarrow{\text{prod}_k} H_k$$

with $H_0 = H$, where, for $i = 1, \dots, k$, $\text{prod}_i = \sum_{e \in B} \text{prod}(e)$, and $H_i \supseteq H(e)_i$ for $e \in B$.² Moreover, for $i = 0, \dots, k$, $H_i = \text{REPLACE}(H, \text{repl}_i)$ and $\text{repl}_i : B \rightarrow \mathcal{H}_C$ is defined by $\text{repl}_i(e) = H(e)_i$ for all $e \in B$.

¹ DER denotes the class of all (parallel) derivations.

² Let, for $e \in B$, $\text{prod}(e) : B(e) \rightarrow P$ be a mapping. Then $\sum_{e \in B} \text{prod}(e)$ denotes the mapping $\sum_{e \in B} \text{prod}(e) : \sum_{e \in B} B(e) \rightarrow P$ defined by $(\sum_{e \in B} \text{prod}(e))(e') = \text{prod}(f)(e')$ for $e' \in B(f)$ and $f \in B$.

Notation

The resulting derivation is denoted by $EMBED(H, fibre)$.

Remarks

1. There is a slightly generalized version of the Joint-Embedding Lemma saying that whenever $type(H(e)_0) = type(e)$, then there is a derivation $H_0 \xRightarrow{prod_1} H_1 \xRightarrow{prod_2} \dots \xRightarrow{prod_k} H_k$, where, for $i = 1, \dots, k$, $prod_i = \sum_{e \in B} prod(e)_i$, and, for $i = 0, \dots, k$, $H_i = REPLACE(H, repl_i)$ and $repl_i : B \rightarrow \mathcal{H}_C$ is defined by $repl_i(e) = H(e)$, for all $e \in B$.
2. The common length of the derivations composed by $EMBED$ is not a serious restriction because each derivation can be lengthened by dummy steps.
3. For a one-element set B , the Joint-Embedding Lemma is a reformulation of the Embedding Lemma in Kreowski [Kr 77b] for the case of hyperedge replacement.

Proof

The Joint-Embedding Lemma is proved by induction on the length of the derivations. The basis, $k = 0$, obviously holds. $repl_0(e) = H(e)_0 = e^*$ for $e \in B$ implies $H_0 = H = REPLACE(H, repl_0)$. Assume now that the statement is true for some $k \geq 0$. Let, for $e \in B$, $H(e)_0 \xRightarrow{prod(e)_1} \dots \xRightarrow{prod(e)_k} H(e)_k \xRightarrow{prod(e)_{k+1}} H(e)_{k+1}$ be a derivation of length $k + 1$ with $H(e)_0 = e^*$. Separating the last derivation step for each derivation, we get derivations of length k . By inductive hypothesis, the derivations can be embedded into H yielding a derivation $H_0 \xRightarrow{prod_1} \dots \xRightarrow{prod_k} H_k$, with $H_0 = H$ where, for $i = 1, \dots, k$, $prod_i = \sum_{e \in B} prod(e)_i$. Moreover, for $i = 0, \dots, k$, $H_i = REPLACE(H, repl_i)$ where, for $e \in B$, $repl_i : B \rightarrow \mathcal{H}_C$ is given by $repl_i(e) = H(e)$. In particular, we have $H_k = REPLACE(H, repl_k)$ where, for $e \in B$, $repl_k : B \rightarrow \mathcal{H}_C$ is given by $repl_k(e) = H(e)_k$. Consider now the direct derivations $H(e)_k \xRightarrow{} H(e)_{k+1}$ through the production bases $prod(e)_{k+1} : B(e)_{k+1} \rightarrow P$ ($e \in B$). Since, for $e \in B$, $prod(e)_{k+1} : B(e)_{k+1} \rightarrow P$ is a production base in $H(e)_k$ and $\sum_{e \in B} B(e)_{k+1} \subseteq \sum_{e \in B} E_{H(e)_k} \subseteq E_{H_k}$, $\sum_{e \in B} prod(e)_{k+1} : \sum_{e \in B} B(e)_{k+1} \rightarrow P$ is a production base in H_k . Hence, there is a direct derivation $H_k \xRightarrow{} H_{k+1}$ through $prod_{k+1} = \sum_{e \in B} prod(e)_{k+1}$. We can assume that

$$\bullet H_{k+1} = REPLACE(H_k, repl'_{k+1})$$

where, for $e' \in \sum_{e \in B} B(e)_{k+1}$, $repl'_{k+1}(e') = rhs(prod_{k+1}(e'))$. Moreover, for $e \in B$,

$$\bullet H(e)_{k+1} = REPLACE(H(e)_k, repl(e)_{k+1})$$

where, for $e' \in B(e)_{k+1}$, $repl(e)_{k+1}(e') = rhs(prod(e)_{k+1}(e'))$. Therefore, we have

$$\begin{aligned} H_{k+1} &= REPLACE(H_k, repl'_{k+1}) \\ &= REPLACE(REPLACE(H, repl_k), repl'_{k+1}) \\ &= REPLACE(H, repl_{k+1}) \end{aligned}$$

where $repl_{k+1} : B \rightarrow P$ is given by $repl_{k+1}(e) = REPLACE(repl_k(e), repl(e)'_{k+1}) = REPLACE(H(e)_k, repl(e)_{k+1}) = H(e)_{k+1}$. This completes the inductive proof. $\square$

2.4 Context-Freeness Lemma

1. Let der be a derivation from H to H' . Then there is a set $B \subseteq E_H$ such that the mapping $fibre : B \rightarrow DER$ given by $fibre(e) = RESTRICT(der, e^*)$ for all $e \in B$ allows to reconstruct the derivation der as $EMBED(H, fibre)$, i.e., (up to isomorphisms³)

$$EMBED(H, fibre) = der.$$

2. Let $H \in \mathcal{H}_C$, $B \subseteq E_H$, and $fibre : B \rightarrow DER$. Then the fibres can be extracted from the embedding derivation $EMBED(H, fibre)$, i.e., for all $e \in E_H$, (up to isomorphism)

$$RESTRICT(EMBED(H, fibre), e^*) = fibre(e).$$

Remark

Roughly speaking, the first statement of the Context-Freeness Lemma says that derivations in hyperedge-replacement grammars can be decomposed into "thin fibres" (where each starts from the subhypergraph induced by a hyperedge) without losing information.

Proof

1. Let $H = H_0 \xRightarrow{prod_1} \dots \xRightarrow{prod_k} H_k$ be a derivation, where, for $i = 1, \dots, k$, $prod_i : B_i \rightarrow P$ is a production base in H_{i-1} . Choose $B = E_H$. Then, for $e \in B$, there is a derivation $e^* = H(e)_0 \xRightarrow{prod(e)_1} \dots \xRightarrow{prod(e)_k} H(e)_k$ where, for $i = 1, \dots, k$, $prod(e)_i$ is the restriction of $prod_i : B_i \rightarrow P$ to $B(e)_i = B_i \cap E_{H(e)_{i-1}}$ and $H(e)_i \subseteq H_i$. Joint embedding of these derivations into H yields a derivation $H'_0 \xRightarrow{prod'_1} \dots \xRightarrow{prod'_k} H'_k$ with $H'_0 = H$ where, for $i = 1, \dots, k$, $prod'_i = \sum_{e \in B} prod(e)_i$. Then we have $H'_0 = H = H_0$ and, if $H'_i = H_i$ and $i < k$, then

$$\begin{aligned} prod'_{i+1} &= \sum_{e \in B} prod(e)_{i+1} \\ &= \sum_{e \in B} prod_{i+1}|_{B_{i+1} \cap E_{H(e)_i}} \\ &= prod_{i+1}|_{B_{i+1} \cap \sum_{e \in B} E_{H(e)_i}} \\ &= prod_{i+1}|_{B_{i+1} \cap E_{H_i}} \\ &= prod_{i+1}. \end{aligned}$$

[Note that, by choice of B , $E_{H_i} = (E_H - B) + \sum_{e \in B} E_{H(e)_i} = \sum_{e \in B} E_{H(e)_i}$.]

Therefore, $H'_{i+1} = H_{i+1}$ (up to isomorphism). This completes the first part of the proof.

³ Let $H_0 \xRightarrow{prod_1} H_1 \xRightarrow{prod_2} \dots \xRightarrow{prod_k} H_k$ and $H'_0 \xRightarrow{prod'_1} H'_1 \xRightarrow{prod'_2} \dots \xRightarrow{prod'_k} H'_k$ be two derivations, where, for $i = 1, \dots, k$, $prod_i : B_i \rightarrow P$ and $prod'_i : B'_i \rightarrow P$. Then the derivations are said to be equal (up to isomorphisms) if there are isomorphisms $iso_i : H_i \rightarrow H'_i$ ($i = 0, \dots, k$) such that, for $i = 1, \dots, k$, $iso_{i-1_B}(B_i) = B'_i$ and $prod_i(e) = prod'_i(iso_{i-1_B}(e))$ for all $e \in B_i$.

2. Let $H \in \mathcal{H}_C$, $B \subseteq E_H$, and, for $e \in B$, $H(e)_0 \xRightarrow{\text{prod}(e)_1} \dots \xRightarrow{\text{prod}(e)_k} H(e)_k$ be a derivation with $H(e)_0 = e^*$. Moreover, let $H_0 \xRightarrow{\text{prod}_1} \dots \xRightarrow{\text{prod}_k} H_k$ be the joint embedding of the derivations into H and, for $e \in B$, $H(e)_0 \xRightarrow{\text{prod}(e)_1'} \dots \xRightarrow{\text{prod}(e)_k'} H(e)_k'$ with $H(e)_0' = e^*$ be the restriction to e^* . Then we have $H(e)_0' = e^* = H(e)_0$ and, if $H(e)_i' = H(e)_i$, and $i < k$, then we have

$$\begin{aligned} \text{prod}(e)_{i+1}' &= \text{prod}_{i+1}|_{B_{i+1} \cap E_{H(e)_i'}} \\ &= \text{prod}_{i+1}|_{B_{i+1} \cap E_{H(e)_i}} \\ &= \text{prod}_{i+1}|_{B(e)_{i+1}} \\ &= \text{prod}(e)_{i+1}. \end{aligned}$$

Therefore, $H(e)_{i+1}' = H(e)_{i+1}$ (up to isomorphism). This completes the second part of the proof. $\square$

3. Derivation Trees

The systematic use of trees to illustrate derivations is an important device that greatly sharpens our intuition. Since we use this concept in formal proofs, it is necessary to have precise definitions. We introduce the notion of derivation trees and discuss the relationship between derivation trees and derivations. Since many derivations may correspond to the same derivation tree, we identify (up to isomorphism) one of these as *canonical*, and thus get a one-to-one correspondence between derivation trees and canonical derivations.

In the following, derivation trees are introduced as labeled hypergraphs. Their hyperedges are labeled by right-hand sides of productions and their nodes are labeled by pairs consisting of a symbol and a hyperedge of the grammar which is labeled by this symbol. The symbols are relevant for further derivations; the occurrence of a hyperedge in a right-hand side is relevant, if the result of the derivation tree shall be determined.

3.1 Definition (derivation trees)

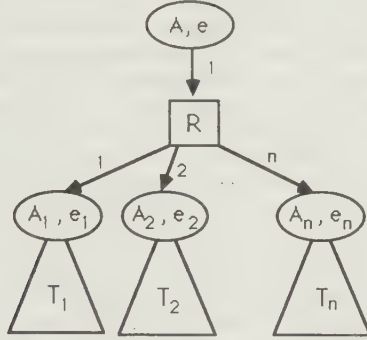
Let P be a set of productions¹. Then a labeled hypergraph T (whose underlying graph is a tree) is a *derivation tree* over P if

- (1) Every node of T is labeled by a pair (A, e) , where $A \in C$ and e is an A -labeled hyperedge occurring in a right-hand side of a production of P .
- (2) Every hyperedge of T is of type $(1, n)$ (with $n \in \mathbb{N}$) and is labeled by the right-hand side of a production of P .
- (3) Every node is the source of at most one hyperedge.
- (4) If a node v in T with label (A, e) is the source of a hyperedge with label R and $v_1, \dots, v_n$ are the targets of the hyperedge with the labels $(A_1, e_1), \dots, (A_n, e_n)$, respectively, then (A, R) is a production of P and $e_1, \dots, e_n$ are the hyperedges of R with the labels $A_1, \dots, A_n$, respectively.

¹ For simplicity, we assume that P is *typed* with respect to C , i.e., there is a function $ltype : C \rightarrow \mathbb{N} \times \mathbb{N}$ such that, for all $R \in rhs(P)$ and all $e \in E_R$, $ltype(l_R(e)) = type(e)$. In this case, A^* determines a $ltype(A)$ -handle induced by $A \in C$. Let us mention that, for each set P , a typed version P' can be considered: Define $P' = \{(A', R') | (A, R) \in P\}$ where, for $(A, R) \in P$, $A' = A_{ltype(R)}$ and $R' = (V_R, E_R, s_R, t_R, l', begin_R, end_R)$ with $l'(e) = l_R(e)_{ltype(e)}$ for $e \in E_R$. We obtain P from P' if we neglect the indices of the labels.

Remark

A derivation tree T over P has the following shape.



The node without incoming tentacles is said to be the root of T and is denoted by $root(T)$. Nodes without outgoing tentacles are called leaves; the set of all leaves of T is denoted by $leaves(T)$.

We shall see that a derivation tree is a natural description of a derivation. If we “join” together all right-hand sides of productions labeling the hyperedges of the derivation tree, we get a hypergraph.

3.2 Definition (result of a derivation tree)

Let $TREE(P)$ be the set of derivation trees over P . Then a mapping $result : TREE(P) \rightarrow \mathcal{H}_C$ can be recursively defined as follows:

- (1) $result(T) = A^*$ for all derivation trees consisting of a single node with label (A, e) .
- (2) $result(T) = REPLACE(R, repl)$
 where $repl : E_R \rightarrow \mathcal{H}_C$ and $repl(e) = result(T(v))$ for all $e \in E_R$ if the hyperedge outgoing from the root is labeled with R , and, for $e \in E_R$, $T(v)$ denotes the subtree of T induced by the node v with label $(l_R(e), e)$.

We shall now show that a derivation tree is an adequate representation for derivations by showing that for every derivation of a hypergraph H through P there is a derivation tree over P with result H , and conversely.

3.3 Lemma

Let P be a set of productions, $A \in C$, and $A^* \xRightarrow{*} H$ be a derivation through P . Then there is a derivation tree T over P such that $result(root(T)) \cong A^*$ and $result(T) \cong H$.

Proof

Suppose that $A^* \xRightarrow{*} H$. We must show that there is a derivation tree T with $result(root(T)) \cong A^*$ and $result(T) \cong H$. If $A^* \xRightarrow{*} H$ is a zero-step derivation, then $A^* \cong H$ and there is a derivation tree T consisting of a single node with label (A, e) . By

definition of *result*, $\text{result}(T) = A^* \cong H$. Now, assume that for any $A \in C$ if $A^* \xRightarrow{*} H$ is a derivation of fewer than $k+1$ steps, then there is a derivation tree with result H (the root of which is labeled by (A, e)). Suppose that $A^* \xRightarrow{*} H$ is a derivation of $k+1$ steps. Let the first step be $A^* \Rightarrow R'$. If the first derivation step is a dummy, then $A^* \cong R'$, $R' \xRightarrow{*} H$ can be modified into $A^* \xRightarrow{*} H$, and, by inductive hypothesis, there is a derivation tree with the desired properties. Otherwise, $R' \cong R$ for some production $(A, R) \in P$. Let $e_1, \dots, e_n$ be the hyperedges in R with labels $A_1, \dots, A_n$, respectively. Consider, for $i = 1, \dots, n$, the fibre $l_{R'}^*(e_i) \xRightarrow{*} H(e_i)$. By the inductive hypothesis, for $i = 1, \dots, n$, there is a derivation tree T_i such that $\text{result}(\text{root}(T_i)) \cong l_{R'}^*(e_i)$ and $\text{result}(T_i) \cong H(e_i)$. We begin by constructing a derivation tree with a root labeled (A, e) (for some e), a hyperedge labeled R , and n leaves labeled $(A_1, e_1), \dots, (A_n, e_n)$. Each node with label (A_i, e_i) is replaced by the tree T_i . In this way, we obtain a tree T with $\text{result}(T) = \text{REPLACE}(R, \text{repl})$ where $\text{repl}(e_i) = \text{result}(T_i)$ for $i = 1, \dots, n$. By the inductive hypothesis, $\text{result}(T_i) = H_i$ for $i = 1, \dots, n$. By the Context-Freeness Lemma, $H = \text{REPLACE}(R, \text{repl})$ where $\text{repl}(e_i) = H_i$ for $i = 1, \dots, n$. Thus, $\text{result}(T) = \text{REPLACE}(R, \text{repl}) = H$. $\square$

We will now obtain the converse of Lemma 3.3. That is, for every derivation tree over P there is at least one derivation through P . To do so we introduce a few more terms.

3.4 Definition (dissection of derivation trees)

1. Let T be a derivation tree over P . Then a *cut* of T is a non-empty set $C \subseteq V_T$ of nodes of T such that
 - (a) No two nodes in C are on the same path in T , and
 - (b) No other node of T can be added to C without violating (a).
2. A sequence $D = C_0, C_1, \dots, C_k$ of cuts of T is called a *dissection* of T if
 - (a) C_0 contains only the root of T ,
 - (b) C_{i+1} is obtained from C_i by replacing some non-leaves in C_i by their direct descendants, for $0 \leq i < k$.
 - (c) C_k is the set of leaves of T .
3. A dissection $D = C_0, C_1, \dots, C_k$ of T is said to be *canonical* if, for all i with $0 \leq i < k$, C_{i+1} is obtained from C_i by replacing all non-leaves of C_i by their direct descendants.

Remarks

1. Each cut $C = \{v_1, \dots, v_n\}$ of T determines a “decomposition” of T into a derivation tree $T' \subseteq T$, denoted by $T|C$, with $\text{root}(T') = \text{root}(T)$ and $\text{leaves}(T') = C$ and a collection of derivation trees, $T_1, \dots, T_n \subseteq T$, with $\text{root}(T_i) = v_i$ for $i = 1, \dots, n$.
2. Each dissection $C_0, C_1, \dots, C_k$ of T determines a sequence $T_0, T_1, \dots, T_k$ of derivation trees such that $T_0 \subseteq T_1 \subseteq \dots \subseteq T_k$.

3.5 Lemma

Let T be a derivation tree over P . Then, for each dissection $D = C_0, C_1, \dots, C_k$ of T , there is a derivation $H_0 \Rightarrow H_1 \Rightarrow \dots \Rightarrow H_k$ through productions of P such that $\text{result}(T|C_i) = H_i$ for $0 \leq i \leq k$.

Proof

1. Let T be a derivation tree with result H and $D = C_0, C_1, \dots, C_k$ be a dissection of T . We prove, by induction on the number of cuts in D , that there is a derivation to $H_0 \Rightarrow H_1 \Rightarrow \dots \Rightarrow H_k$ in P having the desired properties. If there is only one cut C_0 in the sequence D , then the tree must consist of a single node. In this case, there is a derivation (of length 0) from H_0 to H_0 where $H_0 = \text{result}(T) = \text{result}(T|C_0)$. Now suppose that the result is true for dissections with up to k cuts. Also suppose that T is a derivation tree and $D = C_0, C_1, \dots, C_{k+1}$ is a dissection of T for some $k > 0$. If $C_k = C_{k+1}$, then $T|C_k = T|C_{k+1}$, $D' = C_0, \dots, C_k$ is a dissection with up to k cuts, and, by inductive hypothesis, there is a derivation $H_0 \Rightarrow H_1 \Rightarrow \dots \Rightarrow H_k$ with the desired properties. Moreover, there is a derivation $H_k \Rightarrow H_k$ through the empty base. Otherwise, by definition of dissections, C_{k+1} is obtained from C_k by replacing some non-leaves, say, $v_1, \dots, v_n$, by its direct descendants. Let $(A_1, e_1), \dots, (A_n, e_n)$ be the labels of these nodes and $R_1, \dots, R_n$ be the labels of the outgoing hyperedges. $e_1, \dots, e_n$ correspond to hyperedges $e'_1, \dots, e'_n$ in H_k . Therefore, there is a production base $\text{prod} : \{e'_1, \dots, e'_n\} \rightarrow P$ with $\text{prod}(e'_i) = (A_i, R_i)$ and a direct derivation $H_k \Rightarrow H_{k+1}$ where $H_{k+1} = \text{REPLACE}(H_k, \text{repl})$ and $\text{repl}(e'_i) = R_i$ (for $i = 1, \dots, n$). On the other hand, $\text{result}(T|C_{k+1}) = \text{REPLACE}(\text{result}(T|C_k), \text{repl}) = \text{REPLACE}(H_k, \text{repl}) = H_{k+1}$. [Note that the associativity of REPLACE is used]. $\square$

Remark

The canonical dissection of a derivation tree T yields a *canonical derivation* (in the sense of Kreowski, see, e.g., [Kr 87b]) with result $\text{result}(T)$.

We can combine Lemma 3.3 and 3.5 into the following theorem.

3.6 Theorem

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. Then, for all $A \in N$, there is a derivation $A^* \xRightarrow{*} H$ in HRG if and only if there is a derivation tree T for HRG with $\text{result}(\text{root}(T)) \cong A^*$ and $\text{result}(T) \cong H$.

Proof

The proof follows immediately from Lemmas 3.3 and 3.5. $\square$

4. Bibliographic Note

Parallel derivations were introduced by Ehrig and Kreowski ([EK 75], [Kr 77a+b]); the constituting parts of the Sequentialization Theorem can be found, e.g., in [Kr 77a+b] and [Eh 79]. The Context-Freeness Lemma was first formulated and proved for edge-replacement grammars (see [HK 83+85]); a formulation for hyperedge-replacement grammars can be found in [HK 87b]. Finally, derivation trees in the sense of Definition 3.1 are closely related to rule trees as introduced and investigated by Kreowski [Kr 86+87a].

III. Characterizations of HRL's

In this thesis, we view (hyper)graph grammars mainly as language-generating devices. Therefore it is very interesting to have characterizations of the class of languages that can be generated by hyperedge-replacement grammars.

In this chapter, two well-known characterizations of context-free string languages are generalized and adapted to hyperedge-replacement grammars. Gruska [Gr 71] and Yntema [Yn 71] developed a context-free analogon to Kleene's characterization of regular languages. Ginsburg and Rice [GR 62] proved that context-free languages are the least fixed points of their generating productions (considered as a system of language equations). For both results we present hyperedge-replacement-grammar versions providing alternative mechanisms for generating hyperedge-replacement languages.

The chapter is organized as follows. In Section 1, we define some operations, called *substitution* and *iterated substitution*, and show that hyperedge-replacement languages are closed under these operations. The operations turn out to be essential with respect to Section 2 as well as to Section 3. In Section 2, a Kleene-Type Characterization of hyperedge-replacement languages is given. Roughly speaking, it says that the class of all hyperedge-replacement languages is the smallest class of hypergraph languages which contains all finite (homogeneous) languages and is closed under substitution and iterated substitution. Finally, in Section 3, a Fixed-Point Theorem is stated saying that hyperedge-replacement languages are the least fixed points of their generating productions (considered as a system of language equations).

1. Closure Properties

In this section, we consider some operations that preserve hyperedge-replacement languages (shortly denoted as HRL's). The operations are useful not only in constructing or proving that certain languages are hyperedge-replacement languages, but, furthermore, in proving certain languages not to be hyperedge-replacement languages. A given language L can be shown not to be a HRL by constructing from L a language that is not a HRL using operations that preserve HRL's.

The class of hyperedge replacement languages has the interesting property that it is closed under substitution in the following sense. For each symbol A in the alphabet of some hyperedge-replacement language L , let $SUB(A)$ be a particular hyperedge-replacement language. Suppose that we replace each hypergraph H in L with hyperedges $e_1, \dots, e_n$ and labels $A_1, \dots, A_n$, respectively, by hypergraphs $R_1, \dots, R_n$, where R_i is an arbitrary hypergraph in $SUB(A_i)$ ($i = 1, \dots, n$). Then the result is always a hyperedge-replacement language.

1.1 Definition (X -substitution)

Let $X \subseteq C$ be a finite set of symbols. Then a mapping $SUB : X \rightarrow \mathcal{L}_C^1$ is said to be an X -substitution. $SUB : X \rightarrow \mathcal{L}_C$ is extended to languages $L \in \mathcal{L}_C$ by defining

$$SUB(L) = \bigcup_{H \in L} \{REPLACE(H, repl) \mid repl(e) \in SUB(l_H(e)) \text{ for } e \in E_{H,X}\}$$

where, for $H \in \mathcal{H}_C$, $E_{H,X}$ denotes the set $\{e \in E_H \mid l_H(e) \in X\}$ and $repl : E_{H,X} \rightarrow \mathcal{H}_C$ is a base in H , i.e., $repl$ satisfies $type(repl(e)) = type(e)$ for all $e \in E_{H,X}$.

Remark

Each X -substitution $SUB : X \rightarrow \mathcal{L}_C$ induces a set of productions over X , denoted by $P(SUB)$, which is given by $P(SUB) = \{(A, R) \mid A \in X \text{ and } R \in SUB(A)\}$.

Using similar arguments as in the string case we get the following closure property.

1.2 Theorem (closure under X -substitution)

For all finite sets $X \subseteq C$, the hyperedge-replacement languages over C are closed under X -substitution. That is, if L and $SUB(A)$ ($A \in X$) are hyperedge-replacement languages, then $SUB(L)$ is a hyperedge-replacement language, too.

Proof

Let L and $SUB(A)$ ($A \in X$) be hyperedge-replacement languages over C and $T \subseteq C$ the finite set of symbols used. Let $HRG = (N, T, P, Z)$ and $HRG(A) = (N(A), T, P(A), Z(A))$ ($A \in X$) be typed and well-formed hyperedge-replacement grammars generating L and

¹ For a set C , $\mathcal{L}_C$ denotes the class of all hypergraph languages over C .

$SUB(A)$, respectively. Without loss of generality, assume that the nonterminals of HRG and the $HRG(A)$'s are disjoint. Construct a hyperedge-replacement grammar $HRG' = (N', T, P', Z)$ with $N' = N + \sum_{A \in X} N(A)$ and $P' = \dot{P} + \sum_{A \in X} P(A)$ where $\dot{P} = \{(A, \hat{R}) | (A, R) \in P\}$ and, for $R \in \mathcal{H}_C$, $\hat{R}$ is formed from R by substituting $Z(A)$, the axiom of $HRG(A)$, for each instance of an A in X appearing in R .²

We will show that $SUB(L) = L(HRG')$.

First, let $H' \in SUB(L)$. Then, H' can be written as $H' = REPLACE(H, repl)$ where $H \in L$ and $repl : E_{H,X} \rightarrow \mathcal{H}_C$ is a base in H with $repl(e) \in SUB(l_H(e))$. Obviously, there is a derivation $Z \xRightarrow{*} H$ of H in HRG and a corresponding derivation $Z \xRightarrow{*} \hat{H}$ in HRG' where $\hat{H}$ is formed from H by substituting $Z(A)$ for each instance of an $A \in X$ appearing in H . Moreover, for each $e \in E_{H,X}$, there is a derivation $Z(l_H(e)) \xRightarrow{*} repl(e)$ in $HRG(l_H(e))$. Joint embedding of these derivations into $\hat{H}$ yields a derivation $\hat{H} \xRightarrow{*} \hat{H}'$ in HRG' where $\hat{H}' = REPLACE(\hat{H}, repl)$. Since the $Z(l_H(e))$'s are singletons and $type(e) = type(Z(l_H(e)))$ for all $e \in E_{H,X}$, H and $\hat{H}$ only differ in the labels of the hyperedges. Therefore, $REPLACE(\hat{H}, repl) = REPLACE(H, repl)$ and $\hat{H}' = H'$. Now $Z \xRightarrow{*} \hat{H} \xRightarrow{*} H'$ forms a derivation of H' in HRG' . Thus, $H' \in L(HRG')$.

For the converse, let $H' \in L(HRG')$ and $Z \xRightarrow{*} H'$ be a derivation of H' in HRG' . Then the derivation can be transformed in a suitable way. We make use of the fact that "sequentially independent" derivation steps can be interchanged. In the special case of hyperedge replacement, two consecutive steps are sequentially independent if and only if the hyperedge which is replaced in the second step is not just created in the first step. Consequently, derivation steps using productions corresponding to those of HRG are sequentially independent of previous derivation steps using productions of the $HRG(A)$'s ($(\sum N(A) \cup T) \cap N = \emptyset$). Hence, there is a derivation of the form

$$Z \xRightarrow{\dot{P}} \hat{H} \xRightarrow{\sum_{P(A)}} H'$$

where $\hat{H}$ may be obtained from some $H \in \mathcal{H}_T$ by substituting $Z(A)$ for each instance of an $A \in X$ appearing in H . For the derivation $Z \xRightarrow{*} \hat{H}$ in HRG' , there is a corresponding derivation $Z \xRightarrow{*} H$ in HRG ; hence $H \in L$. Moreover, for each $e \in E_{\hat{H},X}$ ($= E_{H,X}$), there is a derivation $Z(l_H(e)) \xRightarrow{*} H'(e)$ in $HRG(l_H(e))$ (see the generalized version of the Restriction Lemma II.2.2), i.e., $H'(e) \in L(HRG(l_H(e)))$. By the Context-Freeness Lemma II.2.4, $H' = REPLACE(H, repl)$ with $repl(e) = H'(e) \in L(HRG(l_H(e)))$ ($e \in E_{H,X}$). Since, for $e \in E_{H,X}$, $L(HRG(l_H(e))) = SUB(l_H(e))$, we get $H' \in SUB(L)$. $\square$

Remarks

1. An X -homomorphism $HOM : X \rightarrow \mathcal{L}_C$ is an X -substitution such that $HOM(A)$ contains a single hypergraph for each A . Since an X -homomorphism is a special type of

² Without loss of generality, we can assume that for each right-hand side R of a production in HRG and each hyperedge $e \in E_R$ with $l_R(e) \in X$, $type(e) = type(Z(l_R(e)))$. Otherwise, e cannot be replaced by a hypergraph of the language $SUB(l_R(e))$; in this case, the production with the right-hand side R is useless with respect to the generation of $SUB(L)$ and can be removed from the production set of HRG .

X -substitution, hyperedge-replacement languages are closed under X -homomorphisms.

2. Hyperedge-replacement languages are closed under *concatenation*. For making this precise, we define a concatenation operation on hypergraph languages L which are *homogeneous* in the sense that, for all $H, H' \in L$, $\text{frame}(H) \cong \text{frame}(H')$. (Hyperedge-replacement languages automatically fulfill this condition.) Roughly, the concatenation $\oplus$ of homogenous hypergraph languages L_1 and L_2 (with respect to an equivalence relation REL on natural numbers), is given by

$$L_1 \oplus L_2 = \{H_1 \oplus H_2 \mid H_1 \in L_1 \text{ and } H_2 \in L_2\}$$

where, for $H_1 \in L_1$, $H_2 \in L_2$, $H_1 \oplus H_2$ is the quotient hypergraph $(H_1 + H_2) / \approx_{REL}$ of $H_1 + H_2$ with respect to the equivalence relation $\approx_{REL}$ induced by REL (see Chapter I, proof of Theorem 4.6). (Note that, in a global way, REL determines how H_1 and H_2 have to be “coupled” in external nodes.)

Now the closure result can be obtained as follows. Let, for $i = 1, 2$, L_i be a hyperedge-replacement language and A_i^* be a handle with label A_i (with a frame corresponding to the frames of the hypergraphs in L_i). Since the finite hypergraph language $\{A_1^* \oplus A_2^*\}$ (resp. $[A_1^* \oplus A_2^*]$) is a hyperedge-replacement language, the closure of hyperedge-replacement languages under $\{A_1, A_2\}$ -substitution implies the closure under the concatenation $\oplus$.

3. Finally, hyperedge-replacement languages are closed under *union* provided that the union yields a homogeneous set. This may be seen as follows: For $i = 1, 2$, let L_i be a hyperedge-replacement language and A_i^* be a handle with label A_i (with a frame corresponding to the frames of the hypergraphs in L_i). If now $L_1 \cup L_2$ is homogeneous, then the finite homogeneous set $\{A_1^*, A_2^*\}$ (resp. $[A_1^*] \cup [A_2^*]$) is a hyperedge-replacement language. Therefore, the closure of hyperedge-replacement languages under $\{A_1, A_2\}$ -substitution implies the closure under union.

1.3 Definition (iterated X -substitution)

Let $SUB : X \rightarrow \mathcal{L}_C$ be an X -substitution. Then the *iterated X -substitution* of SUB , denoted by $SUB^* : X \rightarrow \mathcal{L}_C$, is defined by

$$SUB^*(A) = \bigcup_{n=0}^{\infty} SUB^n(A)$$

where $SUB^0(A) = \emptyset$ and $SUB^{i+1}(A) = SUB^i(SUB(A))$ for $i \geq 0$.

Remarks

1. For each X -substitution $SUB : X \rightarrow \mathcal{L}_C$, SUB^* is an X -substitution, too. Therefore, the extension to languages $L \in \mathcal{L}_C$ is defined. It is straightforward to show that

$$SUB^*(L) = \bigcup_{n=0}^{\infty} SUB^n(L)$$

where $SUB^0(L) = L \cap \mathcal{H}_{C-X}$ and $SUB^{i+1}(L) = SUB^i(SUB(L))$ for $i \geq 0$.

2. For $A \in X$, $L \in \mathcal{L}_C$, and $n \in \mathbb{N}$,

$$SUB^n(A) \subseteq SUB^{n+1}(A) \quad \text{and} \quad SUB^n(L) \subseteq SUB^{n+1}(L).$$

Note that, for $n = 0$, $SUB^0(A) = \emptyset \subseteq SUB^1(A)$ and $SUB^0(L) = L \cap \mathcal{H}_{C-X} \subseteq SUB^1(L)$. For $n \geq 0$, $SUB^{n+1}(A) = SUB^n(SUB(A)) \subseteq SUB^{n+1}(SUB(A)) = SUB^{n+2}(A)$ and $SUB^{n+1}(L) = SUB^n(SUB(L)) \subseteq SUB^{n+1}(SUB(L)) = SUB^{n+2}(L)$.

Before we show that hyperedge-replacement languages are closed under iterated X -substitution, we present a lemma which demonstrates the close relationship between iterated X -substitution and parallel derivations through productions induced by the substitution mapping.

1.4 Lemma (correspondence to parallel derivations)

Let $SUB : X \rightarrow \mathcal{L}_C$ be an X -substitution³ and

$$P(SUB) = \{(A, R) \mid A \in X \text{ and } R \in SUB(A)\}$$

be the set of productions induced by SUB . Then, for $A \in X$ and $L \in \mathcal{L}_C$,

- (1) $SUB^*(A) = \{H \in \mathcal{H}_{C-X} \mid A^* \xrightarrow[P(SUB)]{*} H\},$
- (2) $SUB^*(L) = \bigcup_{H \in L} \{H' \in \mathcal{H}_{C-X} \mid H \xrightarrow[P(SUB)]{*} H'\}.$

Proof

For showing (1), we prove

$$(*) \quad SUB^n(A) = \{H \in \mathcal{H}_{C-X} \mid A^* \xrightarrow[P(SUB)]{n} H\} \text{ for } A \in X$$

by induction on n . Basis step: For $n = 0$, $SUB^0(A) = \emptyset$ and, for all derivations $A^* \xrightarrow{0} H$, $A^* \cong H \notin \mathcal{H}_{C-X}$. Inductive step: Assume that $(*)$ is true for some $n \geq 0$. First, let $A \in X$ and $H \in SUB^{n+1}(A)$. Then, by definition, there is a hypergraph $R \in SUB(A)$ and a mapping $repl$ with $repl(e) \in SUB^n(l_R(e))$ for all $e \in E_{R,X}$. By inductive hypothesis, there are derivations $l_R(e)^* \xrightarrow{n} repl(e)$ through productions of $P(SUB)$. Joint embedding of these derivations into R yields a derivation $R \xrightarrow{n} H'$ where $H' = REPLACE(R, repl) = H$. Moreover, there is a derivation $A^* \Rightarrow R$ through $(A, R) \in P(SUB)$. Hence we have a derivation $A^* \xrightarrow{n+1} H$ through productions of $P(SUB)$. Note that H is in $\mathcal{H}_{C-X}$ because all $repl(e)$ are in $\mathcal{H}_{C-X}$ and all hyperedges in R with label in X are replaced by hypergraphs

³ We assume that $SUB : X \rightarrow \mathcal{L}_C$ is typed with respect to X , i.e., there is a function $ltype : X \rightarrow \mathbb{N} \times \mathbb{N}$ such that, for all $A \in X$, all $H \in SUB(A)$, and all $e \in E_H$ with $l_H(e) \in X$, $ltype(l_H(e)) = type(e)$. In this case, A^* determines a $ltype(A)$ -handle induced by A . Let us mention that, for each X -substitution $SUB : X \rightarrow \mathcal{L}_C$, a typed version SUB' can be constructed such that $SUB'^n(A) = SUB^n(A)$ for $A \in X$: Define $SUB'(A) = \{H' \mid H \in SUB(A)\}$ where, for $H \in \mathcal{H}_C$, $H' = (V_H, E_H, s_H, t_H, l', begin_H, end_H)$ with $l'(e) = l_H(e)_{ltype(e)}$ for $e \in E_H$ with $l_H(e) \in X$ and $l'(e) = l_H(e)$ otherwise. Then SUB' is typed (with respect to X).

in $\mathcal{H}_{C-X}$. Conversely, let $H \in \mathcal{H}_{C-X}$, $A^* \xrightarrow{n+1} H$ be a derivation through productions of $P(SUB)$, and $A^* \Rightarrow R'$ the first derivation step. If this derivation step is a dummy, then $A^* \cong R'$ and the remaining derivation $R' \xRightarrow{n} H$ can be transformed into a derivation $A^* \xRightarrow{n} H$. By inductive hypothesis, $H \in SUB^n(A)$. Since $SUB^n(A) \subseteq SUB^{n+1}(A)$, we get $H \in SUB^{n+1}(A)$. Otherwise, there is a production $(A, R) \in P(SUB)$ such that $R' \cong R$. Without loss of generality, $R' = R$. By the Context-Freeness Lemma II.2.4, there are derivations $l_R(e)^* \xRightarrow{n} repl(e)$, the restrictions of $R' \xRightarrow{n} H$ to $l_R(e)^*$, such that $REPLACE(R, repl) = H$. By inductive hypothesis, $repl(e) \in SUB^n(l_R(e))$ for $e \in E_{R,X}$. Thus, $H \in SUB^{n+1}(A)$. This completes the inductive proof.

Now, (1) directly follows from (*):

$$\begin{aligned} SUB^*(A) &= \bigcup_{n \in \mathbb{N}} SUB^n(A) = \bigcup_{n=0}^{\infty} \{H \in \mathcal{H}_{C-X} \mid A^* \xrightarrow[n]{P(SUB)} H\} \\ &= \{H \in \mathcal{H}_{C-X} \mid A^* \xrightarrow[*]{P(SUB)} H\} \end{aligned}$$

(2) follows from Definition 1.3 and statement (1):

$$\begin{aligned} SUB^*(L) &= \bigcup_{H \in L} \{REPLACE(H, repl) \mid repl(e) \in SUB^*(l_H(e)) \text{ for } e \in E_{H,X}\} \\ &= \bigcup_{H \in L} \{REPLACE(H, repl) \mid l_H(e)^* \xrightarrow[*]{P(SUB)} repl(e) \text{ for } e \in E_{H,X} \text{ and } repl(e) \in \mathcal{H}_{C-X}\} \\ &= \bigcup_{H \in L} \{H' \in \mathcal{H}_{C-X} \mid H \xrightarrow[*]{P(SUB)} H'\} \end{aligned}$$

□

1.5 Theorem (closure under iterated X -substitution)

For all finite sets $X \subseteq C$, the hyperedge-replacement languages over C are closed under iterated X -substitution. That is, if L and $SUB(A)$ ($A \in X$) are hyperedge-replacement languages, then $SUB^*(L)$ is a hyperedge-replacement language, too.

Proof

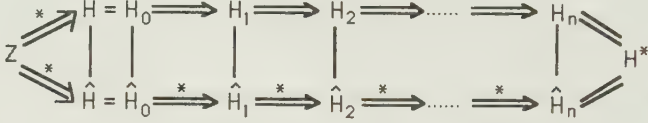
Let L and $SUB(A)$ ($A \in X$) be hyperedge-replacement languages over C and $T \subseteq C$ the finite set of symbols used. Let $HRG = (N, T, P, Z)$ and $HRG(A) = (N(A), T, P(A), Z(A))$ ($A \in X$) be typed and well-formed hyperedge-replacement grammars generating L and $SUB(A)$, respectively. Without loss of generality, assume that the nonterminals of HRG and the $HRG(A)$'s are disjoint. Construct a hyperedge-replacement gr $\bar{a}$. $HRG^* = (N', T', P', Z)$ with $N' = N + \sum_{A \in X} N(A)$, $T' = T - X$, and $P' = \hat{P} + \sum_{A \in X} (A)$ where $\hat{P} = \{(A, \hat{R}) \mid (A, R) \in P\}$, $\hat{P}(A) = \{(A, \hat{R}) \mid (A, R) \in P(A)\}$ ($A \in X$), and, for $R \in \mathcal{H}_C$, $\hat{R}$ is formed from R by substituting $Z(A)$, the axiom of $HRG(A)$, for each instance of an A in X appearing in R .

We will show that $SUB^*(L) = L(HRG^*)$.

First, let $H^* \in SUB^*(L)$. By Lemma 1.4, there is a hypergraph $H \in L$ and a derivation of the form

$$H = H_0 \xrightarrow{P(SUB)} H_1 \xrightarrow{P(SUB)} \dots \xrightarrow{P(SUB)} H_n = H^*$$

For each production $(A, R) \in P(SUB)$, $R \in SUB(A) = L(HRG(A))$, i.e., there is a derivation $Z(A) \xRightarrow{*} R$ through productions of $P(A)$. Therefore, for each direct derivation $H_i \xRightarrow{*} H_{i+1}$ ($i = 0, \dots, n-1$) through $P(SUB)$, there is a derivation $\hat{H}_i \xRightarrow{*} \hat{H}_{i+1}$ through $\sum P(A)$ and a corresponding derivation $\hat{H}_i \xRightarrow{*} \hat{H}_{i+1}$ through $\sum \hat{P}(A)$.

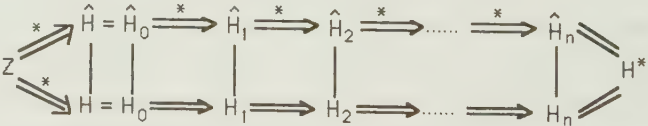


Moreover, for $H \in L = L(HRG)$, there is a derivation $Z \xRightarrow{*} H$ through P and a corresponding derivation $Z \xRightarrow{*} \hat{H}$ through $\hat{P}$. Composing the derivations, one obtains a derivation $Z \xRightarrow{*} \hat{H}_n$ through P' . Since $H_n = H^*$ and $H^* \in \mathcal{H}_{G-X}$, $\hat{H}_n = H_n = H^*$. Hence, $H^* \in L(HRG^*)$.

Conversely, let $H^* \in L(HRG^*)$. Then there is a derivation $Z \xRightarrow{*} H^*$ through productions of P' and $H^* \in \mathcal{H}_{T-X}$. Making use of the fact that sequentially independent derivation steps can be interchanged, the derivation can be transformed into a derivation of the form

$$Z \xRightarrow[\hat{P}]{*} \hat{H} = \hat{H}_0 \xRightarrow[\sum \hat{P}(A)]{*} \hat{H}_1 \xRightarrow[\sum \hat{P}(A)]{*} \dots \xRightarrow[\sum \hat{P}(A)]{*} \hat{H}_n = H^*$$

where, for $i = 0, \dots, n-1$, a start production of P' (i.e., a production whose left-hand side is the label of an axiom $Z(A)$ ($A \in X$)) is applied not until a hypergraph $\hat{H}_i$ is derived corresponding to a hypergraph $H_i \in \mathcal{H}_T$. The derivation $Z \xRightarrow{*} \hat{H}$ through $\hat{P}$ corresponds to a derivation $Z \xRightarrow{*} H$ through P with $H \in \mathcal{H}_T$, i.e., $H \in L(HRG) = L$. For each derivation $\hat{H}_i \xRightarrow{*} \hat{H}_{i+1}$ through $\sum \hat{P}(A)$, there is a derivation $\hat{H}_i \xRightarrow{*} \hat{H}_{i+1} \in \mathcal{H}_T$ through $\sum P(A)$. Consider now, for a hyperedge e in H_i with label $l(e)$ in X , the restriction of $\hat{H}_i \xRightarrow{*} \hat{H}_{i+1}$ to $Z(l(e))$. This yields a derivation of the form $Z(l(e)) \xRightarrow{*} \hat{H}_{i+1}(e) \in \mathcal{H}_T$. Obviously, $\hat{H}_{i+1}(e) \in L(HRG(l(e)))$ (the nonterminals of the $HRG(A)$'s are disjoint), i.e., $\hat{H}_{i+1}(e) \in SUB(l(e))$ and $(l(e), \hat{H}_{i+1}(e)) \in P(SUB)$. Therefore, there is a direct derivation $H_i \xRightarrow{*} \hat{H}_{i+1}$ through $P(SUB)$.



Composing the derivations, one obtains a derivation $H \xRightarrow{*} H_n$ through $P(SUB)$. Moreover, $H_n = \hat{H}_n = H^*$ because H^* does not contain hyperedges with label in X . Since $H \in L$, $H \xRightarrow{*} H^*$ is a derivation through $P(SUB)$, and $H^* \in \mathcal{H}_{T-X}$, Lemma 1.4 implies $H^* \in SUB^*(L)$. $\square$

Considering the operations of intersection and complement, we get (main parts of) the following theorem as an immediate consequence of the connection between hypergraphs and string graphs and the corresponding results for strings.

1.6 Theorem (intersection and complementation)

The hyperedge-replacement languages over C are not closed under intersection and complementation. That is, if L , L_1 and L_2 are hyperedge-replacement languages and $\mathcal{H}_C^L$ and $\mathcal{G}_C^L$ are the sets of all hypergraphs and graphs, respectively, the frames of which correspond to the frames of the hypergraphs in L , then

- the intersection $L_1 \cap L_2$,
- the $\mathcal{H}_C^L$ -complementation $\mathcal{H}_C^L - L = \{H \in \mathcal{H}_C^L \mid H \notin L\}$,
- the $\mathcal{G}_C^L$ -complementation $\mathcal{G}_C^L - L = \{G \in \mathcal{G}_C^L \mid G \notin L\}$,
- the $\mathcal{S}_C$ -complementation $\mathcal{S}_C - L = \{S \in \mathcal{S}_C \mid S \notin L\}$

are not necessarily hyperedge-replacement languages.

Proof

If the hyperedge-replacement languages were closed under intersection (resp. $\mathcal{S}_C$ -complementation), then the context-free string languages were closed under intersection (resp. complementation). If the hyperedge-replacement languages were closed under $\mathcal{G}_C^L$ -complementation, it would follow that the set of all graphs would be a hyperedge-replacement language, contradicting the fact that it cannot be generated by a hyperedge-replacement grammar (see, e.g., Chapter IV, 5.12). Finally, if the hyperedge-replacement languages were closed under $\mathcal{H}_C^L$ -complementation, then the set of all hypergraphs would be a hyperedge-replacement language. But for each hyperedge-replacement language, the number of tentacles a hyperedge may have is bounded. In contrast, for the set of all hypergraphs over C , this number is unbounded. $\square$

Although the class of hyperedge-replacement languages is not closed under intersection, the class is closed under intersection with suitable sets. In Chapter V, 3.9, a general theorem is formulated, saying that whenever L_1 can be generated by a hyperedge-replacement grammar from a class C , and L_2 is a “ C -compatible” hypergraph language, then the intersection $L_1 \cap L_2$ can be generated by a hyperedge-replacement grammar, i.e., $L_1 \cap L_2$ is a hyperedge-replacement language.

2. A Kleene-Type Characterization

Kleene [Kl 56] characterized the regular string languages as the smallest class of string languages which contains all finite string languages and is closed under concatenation and star iteration. Gruska [Gr 71] and Yntema [Yn 71] (independently) generalized this result to context-free string languages using symbol substitution and symbol iteration. We are going to develop a similar characterization for hyperedge-replacement languages. The crucial operations in our characterization are A -substitution and iterated A -substitution which are special cases of substitution and iterated substitution as considered in Sect. 1.

A type of substitution that is of specific interest is that in which a single symbol A is the object of substitution. In this case, we speak about A -substitution and iterated A -substitution where the symbol A refers to the set $X = \{A\}$. Moreover, we will use a notation in which the function value of the substitution mapping SUB , $SUB(A)$, becomes "visible".

2.1 Notations

Let $SUB : \{A\} \rightarrow \mathcal{L}_C$ be an A -substitution and $K, L \in \mathcal{L}_C$. Then $SUB_A(K, L)$ and L^A denote the sets

$$\begin{aligned} SUB_A(K, L) &= SUB(K) \\ L^A &= SUB^*(A) \end{aligned} \quad \text{with } SUB(A) = L.$$

$SUB_A(K, L)$ is said to be the A -substitution in K by L ; L^A is called the iterated A -substitution of L into L .

2.2 Theorem (properties of HRL's)

- (1) Finite, homogeneous hypergraph languages over C are HRL's.
- (2) For all $A \in C$, the class of HRL's over C is closed under A -substitution.
- (3) For all $A \in C$, the class of HRL's over C is closed under iterated A -substitution.

Proof

1. Let $L \in \mathcal{L}_C$ be a finite, homogeneous hypergraph language and $T \subseteq C$ the set of labels used. Then L can be generated by a hyperedge-replacement grammar $HRG = (\{S\}, T, P, Z)$ where S is a symbol not in T , Z is a handle with label S the frame of which corresponds to the frame of some $R \in L$ if L is non-empty and is arbitrary otherwise, and $P = \{(S, choose([R])) | R \in L\}$ where $choose : \{ [H] \mid H \in L \} \rightarrow L$ is a mapping which chooses a hypergraph $choose([R]) \in [R]$ for each class $[R] \subseteq L$. By finiteness of L and choice of one hypergraph of each class of isomorphic hypergraphs, P becomes finite. By homogeneity of L , all productions of P can be applied to Z . Since direct derivations are unique up to isomorphism, exactly the hypergraphs of L can be generated.

2. and 3. follow immediately from the Theorems 1.2 and 1.5 of the previous section. $\square$

On the one hand, hyperedge-replacement languages are closed under symbol substitution and iterated symbol substitution (see Theorems 1.2 and 1.5 as well as Theorem 2.2). On the other hand, we can show that each hyperedge-replacement language can be constructed from finite, homogeneous hypergraph languages by symbol substitution and iterated symbol substitution. This leads to the following Characterization Theorem for hyperedge-replacement languages.

2.3 Theorem (Characterization Theorem)

The class of all hyperedge-replacement languages over C is the smallest class CFL of homogeneous languages satisfying the following conditions:

- (1) if $L \in \mathcal{L}_C$ is finite and homogeneous, then $L \in CFL$;
- (2) if $K, L \in CFL$ and $A \in C$, then $SUB_A(K, L) \in CFL$;
- (3) if $L \in CFL$ and $A \in C$, then $L^A \in CFL$.

Proof

Let L be an arbitrary hyperedge-replacement language over C , $T \subseteq C$ be the finite set of symbols used, and $HRG = (N, T, P, Z)$ be a typed and well-formed hyperedge-replacement grammar generating L . Let $A_1, \dots, A_n$ be an enumeration of the nonterminals of HRG with $A_i \neq A_j$ for $i \neq j$ and $A_1 = l(Z)$. Now the proof is organized in the following way:

- (a) On the one hand, we inductively define hypergraph languages $L_{i,j} \in CFL$ ($1 \leq i \leq j \leq n$) making use of HRG and the order on the nonterminals.
- (b) On the other hand, we define hyperedge-replacement grammars $HRG_1, \dots, HRG_n$ with regard to HRG such that, for $i = 1, \dots, n$, $L(HRG_j) = L_{j,j}^{A_i}$ and $HRG = HRG_1$. Then, for $j = 1, \dots, n$, $L(HRG_j) \in CFL$ and, in particular, $L(HRG) \in CFL$.

Based on these ideas, the proof looks as follows.

- (a) For $1 \leq i \leq j \leq n$, we define the languages $L_{i,j}$ by double downward induction from n :
 - $L_{i,n} = \{R' \in \mathcal{H}_{N \cup T} \mid (A_i, R) \in P \text{ and } R \cong R'\}$ for $1 \leq i \leq n$.
 - $L_{i,j-1} = SUB_{A_i}(L_{i,j}, L_{j,j}^{A_i})$ for $1 \leq i < j \leq n$.

Obviously, the languages $L_{i,n}$ are finite, homogeneous languages and, hence, are in CFL . The languages $L_{i,j-1}$ ($1 \leq i < j \leq n$) are constructed from languages in CFL by symbol substitution and iterated symbol substitution, and, hence, are in CFL (see Theorem 2.2). Hence, all languages $L_{i,j}$ (with $1 \leq i \leq j \leq n$) are in CFL .

(b) Let us now define the hyperedge-replacement grammars HRG_j closely related to $HRG = (N, T, P, Z)$: For $1 \leq j \leq n$, let $HRG_j = (N_j, T_j, P_j, Z_j)$ be the hyperedge-replacement grammar with

$$\begin{aligned} N_j &= N - \{A_i \mid 1 \leq i < j\}, \\ T_j &= \{A_i \mid 1 \leq i < j\} \cup T, \\ P_j &= \{(A, R) \in P \mid A \in N_j\}, \\ Z_j &= A_j^*. \end{aligned}$$

Note that $HRG = HRG_1$ and, hence, $L(HRG) = L(HRG_1)$. To complete the proof of the theorem, it remains to show that $L(HRG_j) = L_{j,j}^{A_i}$ for $1 \leq j \leq n$. This follows from the following claim:

Claim

$$(1) L_{i,j} = \{H \in \mathcal{H}_{T_{j+1}} \mid \exists (A_i, R) \in P_i : R \xrightarrow[\mathcal{P}_{j+1}}^* H\} \quad \text{for } 1 \leq i \leq j \leq n. \quad ^1$$

$$(2) L_{j,j}^A = L(HRG_j) \quad \text{for } 1 \leq j \leq n.$$

Proof of the Claim: We will use an upward induction for $k = n - j$ to n respectively a downward induction for j from n . **Basis step:** Let $k = 0$, that is, $j = n$. Then, for $1 \leq i \leq n$, we have

$$\begin{aligned} L_{i,n} &= \{R' \in \mathcal{H}_{N \cup T} \mid (A_i, R) \in P \text{ and } R \cong R'\} \\ &= \{H \in \mathcal{H}_{T_{n+1}} \mid \exists (A_i, R) \in P_i : R \xrightarrow[\mathcal{P}_{n+1}}^* H\} \end{aligned}$$

because $T_{n+1} = N \cup T$ and $P_{n+1} = \emptyset$. Defining $P(A_n, L_{n,n})$ as $\{(A_n, R) \mid R \in L_{n,n}\}$, using Lemma 1.4, and remembering that $P(A_n, L_{n,n}) = P_n$ (up to isomorphism), we get

$$L_{n,n}^A = \{H \in \mathcal{H}_{C-A_n} \mid A_n^* \xrightarrow[\mathcal{P}(A_n, L_{n,n})}^* H\} = \{H \in \mathcal{H}_{T_n} \mid A_n^* \xrightarrow[\mathcal{P}_n}^* H\} = L(HRG_n).$$

Inductive step: Assume now that (1) and (2) are true for some $k \geq 0$, that is, for some $j \leq n$. Moreover, let $1 \leq i < j$. Then, by Definition of $L_{i,j-1}$, Definition 1.1, Lemma 1.4(1), and the inductive hypothesis,

$$\begin{aligned} L_{i,j-1} &= SUB_{A_j}(L_{i,j}, L_{j,j}^A) \\ &= \{H \in \mathcal{H}_{T_{j+1} - \{A_j\}} \mid \exists G \in L_{i,j} : G \xrightarrow[\mathcal{P}(A_j, L_{j,j})}^* H\} \\ &= \{H \in \mathcal{H}_{T_j} \mid \exists G \in \mathcal{H}_{T_{j+1}}, \exists (A_i, R) \in P_i : R \xrightarrow[\mathcal{P}_{j+1}}^* G \text{ and } G \xrightarrow[\mathcal{P}_j}^* H\} \\ &= \{H \in \mathcal{H}_{T_j} \mid \exists (A_i, R) \in P_i : R \xrightarrow[\mathcal{P}_j}^* H\}. \end{aligned}$$

(where $\mathcal{P}(A_j, L_{j,j})$ denotes the set of productions induced by A_j and $L_{j,j}$.) Let us mention that the last equality holds because — on the one hand — $\mathcal{P}_{j+1} \subseteq \mathcal{P}_j$ and — on the other hand — each derivation of the form $R \xrightarrow[\mathcal{P}_{j+1}}^* G \xrightarrow[\mathcal{P}_j}^* H$ where $G \in \mathcal{H}_{T_{j+1}}$.

In the case $i = j - 1$, we have

$$\begin{aligned} L_{j-1,j-1}^A &= \{H \in \mathcal{H}_{C-A_{j-1}} \mid A_{j-1}^* \xrightarrow[\mathcal{P}(A_{j-1}, L_{j-1,j-1})}^* H\} \\ &= \{H \in \mathcal{H}_{T_{j-1}} \mid A_{j-1}^* \xrightarrow[\mathcal{P}_{j-1}}^* H\} = L(HRG_{j-1}). \end{aligned}$$

[On the one hand, Lemma 1.4(1), properties of $L_{j-1,j-1}$ shown above and the fact that $\mathcal{P}_j \subseteq \mathcal{P}_{j-1}$ are used. On the other hand, it is used that each direct derivation $A_{j-1}^* \Rightarrow H$ through $\mathcal{P}_{j-1}$ with $H \in \mathcal{H}_{T_{j-1}}$ can be transformed into a derivation of the form

$$A_{j-1}^* = H_0 \xrightarrow[\mathcal{P}_{j-1}}^* H_1 \xrightarrow[\mathcal{P}_{j-1}}^* \dots \xrightarrow[\mathcal{P}_{j-1}}^* H_n = H$$

where $H_i \in \mathcal{H}_{T_j}$ and $H_i \xrightarrow[\mathcal{P}_{j-1}}^* H_{i+1}$ is a derivation of the form $H_i \xRightarrow[\mathcal{P}_{j-1}} H_i' \xrightarrow[\mathcal{P}_j}^* H_{i+1}$ ($i = 0, \dots, n-1$).] This completes the inductive proof. $\square$

¹ Assume for the sake of simplicity that $T_{n+1} = N \cup T$ and $P_{n+1} = \emptyset$.

3. A Fixed-Point Theorem

We are going to present a Fixed-Point Theorem for hyperedge-replacement languages. Ginsburg and Rice [GR 62] proved that context-free string languages are the least fixed points of their generating productions (considered as a system of language equations). We present a similar result for hyperedge-replacement languages. The construction of the least fixed points provides an alternative method for generating hyperedge-replacement languages.

3.1 Definition (systems of equations, solutions, and least fixed points)

1. Let $N, T \subseteq C$. A mapping $EQ : N \rightarrow \mathcal{L}_{N \cup T}$ is called a *system of equations* over N . N is said to be the set of *indeterminates* or *unknowns* of EQ .

2. Let $EQ : N \rightarrow \mathcal{L}_{N \cup T}$ be a system of equations over N . An N -substitution $L : N \rightarrow \mathcal{L}_T$ is a *solution* (or *fixed point*) of EQ if, for all $A \in N$,

$$L(A) = L(EQ(A)).$$

3. $L : N \rightarrow \mathcal{L}_T$ is a *least fixed point* of EQ if L is a solution of EQ , and, if L' is any other solution of EQ , then $L(A) \subseteq L'(A)$ for all $A \in N$.

3.2 Lemma (existence and uniqueness)

Every system of equations $EQ : N \rightarrow \mathcal{L}_{N \cup T}$ has a unique least fixed point.

Proof

Let $L(A) = \{H \mid \text{for all solutions } L' \text{ of } EQ : H \in L'(A)\}$ for all $A \in N$. Then it is straightforward to show that L is a solution and that $L(A) \subseteq L'(A)$ for all solutions L' . Thus L is the unique least fixed point of EQ . $\square$

We shall now characterize the least fixed point of a system of equations.

3.3 Lemma (characterization of least fixed points)

Let EQ be a system of equations over N . Then the least fixed point of EQ , denoted EQ^* , is given by

$$EQ^*(A) = \bigcup_{n=0}^{\infty} EQ^n(A)$$

where $EQ^0(A) = \emptyset$ and $EQ^{i+1}(A) = EQ'(EQ^i(A))$ for $i \geq 0$.

Proof

It is straightforward to show that the following set equations are valid:

$$EQ^*(A) = EQ'(EQ^*(A))$$

for all $A \in N$. Thus EQ^* is a solution of EQ .

To show that EQ^* is the least fixed point of EQ , we proceed as follows. Let L be an arbitrary solution of EQ . By induction, we can prove the following.

(*) For all $A \in N$ and all $n \geq 1$, $H \in EQ^n(A)$ implies $H \in L(A)$.

First, let $H \in EQ^1(A)$. Then $H \in EQ(A) \cap \mathcal{H}_T$. Therefore, $H \in L(\{H\}) \subseteq L(EQ(A)) = L(A)$, because L is a solution of EQ . Assume now that for all $A \in N$, $H \in EQ^n(A)$ implies $H \in L(A)$ and consider $H \in EQ^{n+1}(A)$. By definition of EQ^{n+1} , $H \in EQ^n(EQ(A))$. Therefore, $H = REPLACE(R, repl)$ where $R \in EQ(A)$ and $repl(e) \in EQ^n(l_R(e))$ for $e \in E_R$ with $l_R(e) \in N$. By inductive hypothesis, $repl(e) \in L(l_R(e))$ for e with $l_R(e) \in N$. Thus, $H \in L(EQ(A))$. Since L is a solution of EQ , we obtain $H \in L(A)$. This completes the inductive proof.

Let now $A \in N$ and $H \in EQ^*(A)$. Since EQ^* is a solution of EQ , $H \in EQ^*(EQ(A))$, i.e., H can be written as $H = REPLACE(R, repl)$ where $R \in EQ(A)$ and $repl(e) \in EQ^*(l_R(e))$ for all $e \in E_R$ with $l_R(e) \in N$. By definition of EQ^* , for each of these hyperedges $e \in E_R$, there is an integer $n(e) \geq 1$ such that $repl(e) \in EQ^{n(e)}(l_R(e))$. Now (*) implies $H = REPLACE(R, repl) \in L(EQ(A))$. Since L is a solution of EQ , $H \in L(A)$. Thus we can conclude that $EQ^*(A) \subseteq L(A)$ for all $A \in N$. It immediately follows that EQ^* is the least fixed point of EQ . $\square$

Remark

Note that $EQ^* : N \rightarrow \mathcal{L}_T$ is the iterated N -substitution of EQ . As a consequence, we have the following nice situation. If the assigned languages, $EQ(A)$, are hyperedge-replacement languages, then constructed languages $EQ^*(A)$ are hyperedge-replacement languages, too. In particular, if, for all $A \in N$, $EQ(A)$ is finite and homogeneous, then, for all $A \in N$, $EQ^*(A)$ is a hyperedge-replacement language.

There is a proper connection between the least fixed point of the system of equations EQ associated with a hyperedge-replacement grammar HRG and the language family L generated by HRG .

3.4 Definition

Let $HRG = (N, T, P, Z)$ be a typed and well-formed hyperedge-replacement grammar. Then the *system of equations*, $EQ : N \rightarrow \mathcal{L}_{N \cup T}$, associated with HRG is given by $EQ(A) = \{R' \mid (A, R) \in P \text{ and } R \cong R'\}$. The *language family* $L : N \rightarrow \mathcal{L}_T$ generated by HRG is given by $L(A) = \{H \in \mathcal{H}_T \mid A \xrightarrow{*} H\}$.

3.5 Fixed-Point Theorem

Let $HRG = (N, T, P, Z)$ be a typed and well-formed hyperedge-replacement grammar, $EQ : N \rightarrow \mathcal{L}_{N \cup T}$ be the associated system of equations, and $L : N \rightarrow \mathcal{L}_T$ the generated language family. Then L is the least fixed point of EQ .

Proof

Let $HRG = (N, T, P, Z)$ be a typed and well-formed hyperedge-replacement grammar, $EQ : N \rightarrow \mathcal{L}_{N \cup T}$ be the system of equations associated with HRG , and $L : N \rightarrow \mathcal{L}_T$ the

language family generated by HRG . By Lemma 3.3, EQ^* is the least fixed point of EQ . By Lemma 1.4 and the relationships between $P(EQ)$ and P , we obtain

$$EQ^*(A) = \{H \in \mathcal{H}_T \mid A^* \xrightarrow[P(EQ)]{*} H\} = \{H \in \mathcal{H}_T \mid A^* \xrightarrow[P]{*} H\} = L(A)$$

for all $A \in N$. Hence, $EQ^* = L$. Therefore, L is the least fixed point of EQ . $\square$

Remarks

1. The Fixed-Point Theorem for hyperedge-replacement languages generalizes Ginsburg's and Rice's well-known Fixed-Point Theorem for context-free string languages in [GR 62] as well as the Fixed-Point Theorem for context-free graph languages given in [HK 83+85]. A similar theorem for a slightly different notion of hyperedge-replacement grammar has been established by Bauderon and Courcelle [BC 87]. The proof given in [BC 87] makes use of the fact that a set of hypergraphs can be generated by a hyperedge-replacement grammar if and only if it is "equational", whereas, in this thesis, the Fixed-Point Theorem is proved "directly" without use of other results.
2. The construction of least fixed points provides an alternative method for generating hyperedge-replacement languages.

4. Bibliographic Note

The results on hyperedge-replacement languages presented in this chapter (expressed in a slightly different way) may be seen as adaptations of the results for edge-replacement languages given in [HK 85].

IV. Structural Aspects of HRL's

In this chapter, we present a number of necessary conditions for a hypergraph language to be a hyperedge-replacement language. Beside a Pumping Lemma (which is in the spirit of the well-known Pumping Lemma for context-free string languages) and a version of Parikh's Theorem for hyperedge-replacement languages, a so-called Connectivity Theorem and a Minimum-Degree Theorem are presented. These theorems give necessary but not sufficient conditions for sets of hypergraphs generated by hyperedge-replacement grammars. They can be used to show that certain sets of hypergraphs cannot be generated by this kind of grammars.

The chapter is organized as follows: In Section 1, we consider some simplifications of hyperedge-replacement grammars which will be useful in the following. In Section 2, we state a Pumping Lemma which generalizes those Pumping Lemmata known for context-free string and graph languages. It says roughly that each sufficiently large hypergraph belonging to a hyperedge-replacement language decomposes into $FIRST \otimes LINK \otimes LAST$ such that $FIRST \otimes LINK^k \otimes LAST$ for $k \geq 0$ belongs to the given language. The crucial part is to find the proper composition $\otimes$ for hypergraphs. In Section 3, we present a theorem expressing the semilinearity of hyperedge-replacement languages under the Parikh mapping. Section 4 is concerned with graph-theoretic properties — like the connectivity and the minimal degree — of the generated (hyper)graphs. Finally, in Section 5, we apply the theorems to various languages proving that they cannot be generated by a hyperedge-replacement grammar. In particular, we can show that the generative power of hyperedge-replacement grammars depends strongly on the so-called order, which is an upper bound for the number of "tentacles" a non-terminal hyperedge may have (for gripping at their source and target nodes). In other words, the order induces an infinite hierarchy of classes of hypergraph languages, which remains infinite even if one considers graph or string(-graph) languages only. The string(-graph) languages include certain non-context-free languages, and the graph languages of order 2 are just those which can be generated by edge replacement.

1. Simplifications of HRG's

In this section, we consider — similar to context-free string grammars — “reduced” hyperedge-replacement grammars which do not possess “empty productions” or “single productions”. Furthermore, we show that for each typed hyperedge-replacement grammar there is an equivalent reduced one without empty productions and single productions. The proofs look similar to the corresponding proofs in the string case. This is due to the frequent use of the Context-Freeness Lemma (II.2.4).

We first undertake the task of eliminating those symbols from a grammar which are not used: Let $HRG = (N, T, P, Z)$ be a typed hyperedge-replacement grammar. A symbol $A \in N$ turns out to be *irrelevant* (or *useless*) — insofar as the language $L(HRG)$ is concerned — if no terminal hypergraph can be derived from A^* ¹ or if A does not occur in any hypergraph derived from the axiom Z . Useless symbols of this kind can be removed from the grammar HRG without effect on $L(HRG)$.

1.1 Definition (reduced grammars)

A typed hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is said to be *reduced* iff $P = \emptyset$ or, for all $A \in N$,

- (1) there is a terminal hypergraph derivable from A^* , i.e., $A^* \xRightarrow{*} H$ for some $H \in \mathcal{H}_T$,
- (2) there is a hypergraph derivable from Z for which some hyperedge is labeled with A , i.e., $Z \xRightarrow{*} G$ for some $G \in \mathcal{H}_{N \cup T}$ and some $e \in E_G$ with $l_G(e) = A$.

Remark

For simplicity, reducedness is defined for typed grammars, only. More general, we could define it for arbitrary grammars. But this does not seem to be significant because

- (1) handles with same label but different type may behave different with respect to the question “Is there a terminal hypergraph derivable from the handle?” and
- (2) hyperedges with same label but different type may behave different with respect to the question “Is there a production applicable to the hyperedge?”.

1.2 Theorem (Reducedness Theorem)

For each hyperedge-replacement grammar, an equivalent reduced grammar can be constructed.

¹ Let $HRG = (N, T, P, Z)$ be a typed hyperedge replacement grammar and $ltype : N \rightarrow \mathbb{N} \times \mathbb{N}$ the corresponding type function. Then, for $A \in N$, A^* denotes a $ltype(A)$ -handle induced by A .

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. By the Typification Theorem given in I.4.3, we may assume that HRG is typed. Now an equivalent reduced hyperedge-replacement grammar $HRG_{red} = (N_{red}, T, P_{red}, Z)$ is constructed in two steps. First, the set of nonterminals is restricted to those ones from (the handles of) which a terminal hypergraph can be derived. In the second step, the set of remaining nonterminals is restricted to those which are accessible from the axiom of the grammar. [Note that the procedure is similar to the corresponding proof in the string case (cf., e.g., [HU 69], Theorems 4.2+4.3).]

Step 1. Let $HRG = (N, T, P, Z)$ be typed. Then $HRG' = (N', T, P', Z)$ is constructed as follows: Construct the set $N_T = \{A \in N \mid A^* \xRightarrow{*} H \text{ for some } H \in \mathcal{H}_T\}$, take $N' = N_T \cup \{l(Z)\}$, choose $P' = \emptyset$ if $Z \notin N_T$ and $P' \subseteq P$ to be the set of productions involving symbols of $N_T \cup T$ only, otherwise. Surely, HRG' satisfies the property that $P' = \emptyset$ or, if $A \in N'$, then $A^* \xRightarrow{*} H$ for some $H \in \mathcal{H}_T$. Thus, the main tasks of the first part of the proof are to show that the set N_T can be constructed effectively and that the grammars HRG and HRG' are equivalent.

For constructing the set

- $N_T = \{A \in N \mid A^* \xRightarrow{*} H \text{ for some } H \in \mathcal{H}_T\}$,

we are going to define sets N_k for $k \in \mathbb{N}$ recursively:

- $N_0 = \emptyset$,
- $N_{k+1} = N_k \cup \{A \in N \mid (A, R) \in P \text{ and } l_R(e) \in N_k \cup T \text{ for all } e \in E_R\} \quad \text{for } k \geq 0$.

Since $N_k \subseteq N_{k+1} \subseteq N$ for all $k \in \mathbb{N}$ and N is finite, there is some $l \in \mathbb{N}$ such that $N_l = N_{l+1}$. Obviously, all N_k up to the smallest possible l can be constructed effectively. Hence it remains to prove that $N_T = N_l$ which is an immediate consequence of the following two claims.

Claim 1.1: $N_k \subseteq N_T$ for all $k \in \mathbb{N}$.

Proof of Claim 1.1 (by induction on k): The statement is trivially true for $k = 0$. Assume that it is true for k , and let A be in N_{k+1} . If A is also in N_k , the inductive step is trivial. If A is in $N_{k+1} - N_k$, then there is a production $(A, R) \in P$ such that $l_R(e) \in N_k \cup T$ for all $e \in E_R$. By inductive hypothesis, $l_R(e) \in N_T \cup T$ for all $e \in E_R$. Thus, for all $e \in E_R$, we can find a derivation $l_R(e)^* \xRightarrow{*} H(e)$ with $H(e) \in \mathcal{H}_T$. Joint embedding of these derivations into R yields a derivation $R \xRightarrow{*} H$, where $H \in \mathcal{H}_T$. Moreover, there is a direct derivation $A^* \xRightarrow{*} R$ through the production (A, R) . Composing these derivations, we get a derivation $A^* \xRightarrow{*} H$, where $H \in \mathcal{H}_T$, which means that A is in N_T . This proves Claim 1.1.

Claim 1.2: $N_T \subseteq N_l$.

Proof of Claim 1.2: $A \in N_T$ means that there is a derivation $A^* \xRightarrow{*} H$ for some $H \in \mathcal{H}_T$. Thus, we prove the statement

If $A \in N$ and $A^* \xRightarrow{*} H$ for some $H \in \mathcal{H}_T$, then $A \in N_l$.

by induction on the number of derivation steps. The basis, $k = 1$, is trivial; for each one-step derivation $A^* \xRightarrow{1} H$, $H \cong R$ for some $(A, R) \in P$ and $l_R(e) \in T$ for all $e \in E_R$, i.e.,

$A \in N_1 \subseteq N_i$. Assume that the statement is true for k -step derivations, and let $A^* \xrightarrow{k+1} H$ with $H \in \mathcal{H}_T$. Then the derivation decomposes into $A^* \xRightarrow{*} R' \xrightarrow{k} H$ for some $R' \in \mathcal{H}_{N \cup T}$. If the first derivation step is a dummy, then $A^* \cong R'$, and $R' \xrightarrow{k} H$ can be modified into $A^* \xrightarrow{k} H$. By inductive hypothesis, this implies $A \in N_i$. Otherwise, $R' \cong R$ for some production $(A, R) \in P$. Consider now the fibres $l_{R'}(e)^* \xrightarrow{k} H(e)$ of $R \xRightarrow{*} H$ for $e \in E_{R'}$. Then, for all $e \in E_{R'}$, $H(e) \in \mathcal{H}_T$. Using the inductive hypothesis, we get $l_{R'}(e) \in N_i \cup T$ for all $e \in E_{R'}$. Since $R' \cong R$, we obtain $l_R(e) \in N_i \cup T$ for all $e \in E_R$. By definition and choice of l , we get $A \in N_{i+1} = N_i$. This proves Claim 1.2.

To prove $L(HRG') = L(HRG)$, we have to show $L(HRG') \subseteq L(HRG)$ and $L(HRG) \subseteq L(HRG')$. Clearly the first relationship holds, since $P' \subseteq P$. The second relationship follows immediately from the following claim.

Claim 1.3: If $A \in N$ and $A^* \xRightarrow{*} H$ is a derivation in HRG for some $H \in \mathcal{H}_T$, then $A^* \xRightarrow{*} H$ is a derivation in HRG' .

Proof of Claim 1.3 (by induction on the number of derivation steps): The statement is trivially true for one-step derivations. Suppose now that it is true for k -step derivations. Let $A^* \xrightarrow{k+1} H$ be a $(k+1)$ -step derivation in HRG and $H \in \mathcal{H}_T$. Then the derivation is of the form $A^* \xRightarrow{*} R' \xrightarrow{k} H$ for some $R' \in \mathcal{H}_{N \cup T}$. If the first derivation step is a dummy, then $A^* \cong R'$, and $R' \xrightarrow{k} H$ can be modified into $A^* \xrightarrow{k} H$. By inductive hypothesis, $A^* \xrightarrow{k} H$ is a derivation in HRG' . Otherwise, $R' \cong R$ for some production $(A, R) \in P$. Consider now the fibres $l_{R'}(e)^* \xrightarrow{k} H(e)$ for $e \in E_{R'}$. Obviously, $H(e) \in \mathcal{H}_T$ for $e \in E_{R'}$. By inductive hypothesis, the fibres are derivations in HRG' . Thus, $R' \xrightarrow{k} H$ is a derivation in HRG' . By definition of N_T , $A \in N_T$ and $l_{R'}(e) \in N_T \cup T$ for $e \in E_{R'}$. This implies that (A, R) is a production in P' . Hence $A^* \xRightarrow{*} R' \xrightarrow{k} H$ is a derivation in HRG' . This completes the inductive proof.

By typedness — more exactly, by usualness — of HRG , Z is a singleton with $l(Z) \in N$. Therefore, Claim 1.3 implies $L(HRG) \subseteq L(HRG')$. Thus, $L(HRG) = L(HRG')$.

Step 2. Consider $HRG' = (N', T, P', Z)$ as constructed in Step 1. Then we have to remove all symbols which are not accessible from the axiom Z . Therefore, we construct the set $N'_{\text{access}} = \{B \in N' \mid Z \xRightarrow{*} G \text{ with } l_G(e) = B \text{ for some } e \in E_G\}$. Taking $N'' = N'_{\text{access}}$ and $P'' \subseteq P'$ to be the set of productions in P' which involve symbols of $N'' \cup T$, only, we get a hyperedge-replacement grammar $HRG'' = (N'', T, P'', Z)$, which turns out to be reduced. Thus, the main tasks of the second part of the proof are to show that the set N'_{access} can be constructed effectively and that the grammars HRG' and HRG'' are equivalent.

For constructing the set

- $N'_{\text{access}} = \{B \in N' \mid Z \xRightarrow{*} G \text{ with } l_G(e) = B \text{ for some } e \in E_G\}$,

we are going to define sets N'_k for $k \in \mathbb{N}$ recursively as follows:

- $N'_0 = \{l(Z)\}$,
- $N'_{k+1} = N'_k \cup \{B \in N' \mid A \in N'_k, (A, R) \in P', \text{ and } l_R(e) = B \text{ for some } e \in E_R\}$ for $k \geq 0$.

Since the sets N'_k are monotonically increasing subsets of N' and since N' is finite, there is some $l \in \mathbb{N}$ such that $N'_l = N'_{l+1}$. Clearly, all N'_k up to the smallest possible l can be constructed effectively. Hence it remains to prove that $N'_{\text{access}} = N'_l$ which is an immediate consequence of the following two claims.

Claim 2.1: $N'_k \subseteq N'_{\text{access}}$ for all $k \in \mathbb{N}$.

Proof of Claim 2.1 (by induction on k): The statement is trivially true for $k = 0$; in this case, $N'_0 = \{l(Z)\}$ and $Z \xRightarrow{0} Z$ with $l_Z(e) = l(Z)$ for the single hyperedge $e \in E_Z$. Assume now that the statement is true for k , and let B be in N'_{k+1} . If B is also in N'_k , the inductive step is trivial. Otherwise, B is in $N'_{k+1} - N'_k$, which means that there is a symbol $A \in N'_k$ and a production $(A, R) \in P'$ such that $l_R(e) = B$ for some $e \in E_R$. By inductive hypothesis, $A \in N'_{\text{access}}$, i.e., there is a derivation $Z \xRightarrow{*} G$ with $l_G(e) = A$ for some $e \in E_R$. Let e' be such a hyperedge with $l_G(e') = A$. By typedness of HRG' , the production $(A, R) \in P'$ can be applied to the hyperedge e' in G yielding a direct derivation $G \Rightarrow H$ in HRG' . Since R contains a hyperedge with label B , H does it, too. Composing the derivations, we get a derivation $Z \xRightarrow{*} H$ in HRG' with $l_H(e) = B$ for some $e \in E_H$, i.e., $B \in N'_{\text{access}}$. This completes the inductive proof.

Claim 2.2: $N'_{\text{access}} \subseteq N'_l$.

Proof of Claim 2.2: $B \in N'_{\text{access}}$ means that there is a derivation $Z \xRightarrow{*} H$ with $l_H(e) = B$ for some $e \in E_H$. Thus, we prove the statement

If $Z \xRightarrow{*} H$ and $l_H(e) = B$ for some $e \in E_H$, then $B \in N'_l$.

by induction on the number of derivation steps. The basis, $k = 0$, is trivial; for each zero-step derivation $Z \xRightarrow{0} H$, $H \cong Z$, the single hyperedge in H is labeled by $l(Z)$, and $l(Z) \in N'_0 \subseteq N'_l$. Assume now that the statement is true for k -step derivations, and let $Z \xRightarrow{k+1} H$ with $l_H(e) = B$ for some $e \in E_H$. Then the derivation decomposes into $Z \xRightarrow{k} G \Rightarrow H$ for some $G \in \mathcal{H}_{N' \cup T}$. If the considered hyperedge with label B occurs already in G , the inductive step is trivial. If it is produced in the last derivation step, then we can find a hyperedge $e_0 \in E_G$ and a production $(A, R) \in P'$ with $l_G(e_0) = A$ which derives it. By inductive hypothesis, $A \in N'_l$. By definition and choice of l , we get $B \in N'_{l+1} = N'_l$. This completes the inductive proof and proves Claim 2.2.

Finally, we have to show that $L(HRG'') = L(HRG')$. Surely, $L(HRG'') \subseteq L(HRG')$, because $P'' \subseteq P'$. For proving $L(HRG') \subseteq L(HRG'')$ we show the following statement.

Claim 2.3: Each derivation $Z \xRightarrow{*} H$ in HRG' is a derivation in HRG'' , too.

Proof of Claim 2.3 (by induction on the number of derivation steps): The statement is trivially true for zero-step derivations. Suppose now that it is true for derivations of length k , and let $Z \xRightarrow{k+1} H$ be a $(k+1)$ -step derivation in HRG' . Then the derivation is of the form $Z \xRightarrow{k} G \Rightarrow H$. By inductive hypothesis, $Z \xRightarrow{k} G$ is a derivation in HRG'' , too. Moreover, all nonterminals occurring in G or H are accessible from Z , i.e., they are in N'_{access} . Thus all productions applied to G are productions in P'' , i.e., $G \Rightarrow H$ is a direct derivation in HRG'' . Hence, $Z \xRightarrow{k+1} H$ is a derivation in HRG'' . This completes the inductive proof.

Note that $HRG'' = (N'', T, P'', Z)$ is reduced: $P'' = \emptyset$ or, for each $A \in N'' \subseteq N'$, there is a derivation $A^\bullet \xRightarrow{*} H$, where $H \in \mathcal{H}_T$. Moreover, $A \in N''$ means that there is a derivation $Z \xRightarrow{*} G$ in HRG' with $l_G(e) = A$ for some $G \in \mathcal{H}_{N \cup T}$ and some $e \in E_G$. By Claim 2.3, this is a derivation in HRG'' , too. $\square$

We get the following result as corollary of the proof of Theorem 1.2.

1.3 Corollary (Emptiness Problem)

For hyperedge-replacement grammars, the Emptiness Problem is decidable, i.e., there is an algorithm for determining whether the language of a hyperedge-replacement grammar is empty or not.

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar generating L . By the Typification Theorem (see Chapter I, Theorem 4.3), we may assume that HRG is typed. Now we construct the set $N_T = \{A \in N \mid A^\bullet \xRightarrow{*} H \text{ for some } H \in \mathcal{H}_T\}$ as in the proof of Theorem 1.2. Then $L = L(HRG) \neq \emptyset$ if and only if $l(Z) \in N_T$. $\square$

It is often convenient to eliminate productions which neither produce nodes nor hyperedges. Productions of this type are called **empty productions**.

1.4 Definition (empty productions)

A production (A, R) is said to be **empty** if there are no internal nodes and no hyperedges in the right-hand side of the production, i.e., $INT_R = \emptyset$ as well as $E_R = \emptyset$.

Given a hyperedge-replacement grammar $HRG = (N, T, P, Z)$, we are going to eliminate empty productions. However, if $frame(Z)$ is in $L(HRG)$, then clearly it is impossible to have no empty productions.

1.5 Theorem (elimination of empty productions)

For each hyperedge-replacement grammar, an equivalent typed grammar can be constructed in which either

- (a) there are no empty productions, or
- (b) there is exactly one empty production of the form $(l(Z), frame(Z))$ where Z denotes the axiom of the grammar and the label of Z , $l(Z)$, does not appear on the right-hand side of any production.

Notation

Somewhat misleading, typed hyperedge-replacement grammars satisfying the condition (a) or (b) are called **grammars without empty productions**.

Remark

Theorem 1.5 may be seen as an adaptation of the corresponding result for context-free Chomsky grammars to hyperedge-replacement grammars. Thus, the arguments in the proof look similar to those ones used in the string case (see, e.g., [AU 72], Theorem 2.14, [HU 69], Theorem 4.11).

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. By the Typification Theorem (see Chapter I, Theorem 4.3), we may assume that HRG is typed. Then we construct an equivalent hyperedge-replacement grammar $HRG' = (N', T, P', Z')$ without empty productions using more or less the same ideas as in the string case:

- (1) Construct the set $FRAME = \{A \in N \mid A^* \xRightarrow{*} frame(A^*)\}$.
- (2) Construct $HRG' = (N', T, P', Z')$ according to the following rules:
 - (a) If (A, R) is in P and $B \subseteq \{e \in E_R \mid l_R(e) \in FRAME\}$, then $(A, R - B)$ is in P' provided that $R - B$ is not a frame.²
 - (b) If $l(Z) \in FRAME$, add the productions $(S', frame(Z))$ and (S', Z) to P' where S' is a new symbol with $ltype(S') = ltype(Z)$. In this case, choose $N' = N \cup \{S'\}$ and $Z' = S'^*$. Otherwise, choose $N' = N$ and $Z' = Z$.

Obviously, HRG' is a typed hyperedge-replacement grammar without empty productions (in the sense of 1.5). Thus, the main tasks of the proof are to show that the set $FRAME$ can be constructed effectively and that the grammars HRG and HRG' are equivalent.

1. For constructing the set

- $FRAME = \{A \in N \mid A^* \xRightarrow{*} frame(A^*)\}$,

we are going to define sets $FRAME_k$ for $k \in \mathbb{N}$ recursively. A nonterminal A is included in $FRAME_k$ if there is a derivation $A^* \xRightarrow{j} frame(A^*)$ for some $j \leq k$. Note that $A^* \not\xRightarrow{*} frame(A^*)$ for all $A \in N$ so that $FRAME_0$ does not contain any symbol of N .

- $FRAME_0 = \emptyset$,
- $FRAME_{k+1} = FRAME_k \cup \{A \in N \mid (A, R) \in P, INT_R = \emptyset, \text{ and } l_R(e) \in FRAME_k \text{ for all } e \in E_R\} \text{ for } k \geq 0$.

Since $FRAME_k \subseteq FRAME_{k+1} \subseteq N$ for all $k \in \mathbb{N}$ and N is finite, there is some $l \in \mathbb{N}$ such that $FRAME_l = FRAME_{l+1}$. Obviously, all $FRAME_k$ up to the smallest possible l can be constructed effectively. Hence, it remains to prove that $FRAME = FRAME_l$, which is a consequence of the following two claims.

Claim 1.1: $FRAME_k \subseteq FRAME$ for all $k \in \mathbb{N}$.

Proof of Claim 1.1 (by induction on k): The statement is trivially true for $k = 0$. Assume that it is true for k , and let A be in $FRAME_{k+1}$. If A is also in $FRAME_k$, the inductive step is trivial. If A is in $FRAME_{k+1} - FRAME_k$, then there is a production $(A, R) \in P$ such that $INT_R = \emptyset$ and $l_R(e) \in FRAME_k$ for all $e \in E_R$. By inductive hypothesis, $l_R(e) \in FRAME$ for all $e \in E_R$. Thus, we can find a derivation $l_R(e)^* \xRightarrow{*} frame(l_R(e)^*)$ for each $e \in E_R$. Joint embedding of these derivations into R yields a derivation $R \xRightarrow{*} H$ where $H = R - E_R$ (up to isomorphism). Hence $INT_H = INT_R = \emptyset$ and $E_H = \emptyset$, i.e., H is a frame. Now $H \cong frame(A^*)$ and $A^* \xRightarrow{*} H$ can be modified into $A^* \xRightarrow{*} frame(A^*)$. By definition of the set $FRAME$, $A \in FRAME$. This completes the inductive proof.

² Let $H \in \mathcal{H}_C$ and $B \subseteq E_H$. Then $H - B$ denotes the multi-pointed hypergraph $(V_H, E_H - B, s, t, l, begin_H, end_H)$ where s, t , and l are the restrictions of the mappings s_H, t_H , and l_H to the set $E_H - B$, respectively.

Claim 1.2: $FRAME \subseteq FRAME_I$.

Proof of Claim 1.2: $A \in FRAME$ means that there is a derivation $A^* \xRightarrow{*} frame(A^*)$. Thus, we prove the statement

If $A \in N$ and $A^* \xRightarrow{*} frame(A^*)$, then $A \in FRAME_I$.

by induction on the number of derivation steps. The basis, $k = 1$, is trivial; for each one-step derivation $A^* \xRightarrow{1} frame(A^*)$, there is some $(A, R) \in P$ with $R \cong frame(A^*)$, i.e., $A \in FRAME_1 \subseteq FRAME_I$. Assume that the statement is true for k -step derivations, and let $A^* \xRightarrow{k+1} H$ be a $(k+1)$ -step derivation with $H \cong frame(A^*)$. Then the derivation decomposes into $A^* \Rightarrow R' \xRightarrow{k} H$ for some $R' \in \mathcal{H}_{NUT}$. If the first step is a dummy, then $A^* \cong R'$ and $R' \xRightarrow{k} H$ can be modified into $A^* \xRightarrow{k} frame(A^*)$. By inductive hypothesis, this implies $A \in FRAME_I$. Otherwise, $R' \cong R$ for some production $(A, R) \in P$. Since usual hyperedge-replacement grammars preserve nodes as well as terminal hyperedges and since H is a frame, R cannot possess internal nodes or terminal hyperedges. Consider now the fibres $l_{R'}(e)^* \xRightarrow{k} H(e)$ for $e \in E_{R'}$. Then, for all $e \in E_{R'}$, $l_{R'}(e) \in N$ and $H(e) \cong frame(l_{R'}(e))$ because $R' \cong R$ and $H \cong frame(A^*)$. By inductive hypothesis, $l_{R'}(e) \in FRAME_I$ for each $e \in E_{R'}$. Because R' and R are isomorphic, $l_R(e) \in FRAME_I$ for each $e \in E_R$. By definition and choice of l , we get $A \in FRAME_{I+1} = FRAME_I$. This proves Claim 1.2.

2. To show $L(HRG') = L(HRG)$, we can prove Claim 2.1 and Claim 2.2.

Claim 2.1: For each $A \in N$, if $A^* \xRightarrow{*} H$ is a derivation in HRG' , then $H \not\cong frame(A^*)$ and there is a derivation $A^* \xRightarrow{*} H$ in HRG .

Proof of Claim 2.1 (by induction on the number of steps in the derivation): The statement is true for zero-step derivations; in this case $H \cong A^*$, but $H \not\cong frame(A^*)$. Suppose now that it is true for k -step derivations. Let $A^* \xRightarrow{k+1} H$ be a $(k+1)$ -step derivation in HRG' . Then the derivation is of the form $A^* \Rightarrow R'' \xRightarrow{k} H$. If the first derivation step is a dummy, then $A^* \cong R''$ and $R'' \xRightarrow{k} H$ can be modified into $A^* \xRightarrow{k} H$. By inductive hypothesis, $H \not\cong frame(A^*)$ and there is a derivation $A^* \xRightarrow{*} H$ in HRG . Otherwise, a production $(A, R') \in P'$ is applied to A^* yielding to R'' . [Note that $R' \cong R''$]. By construction of P' , there is a production $(A, R) \in P$ and a set $B \subseteq \{e \in E_R \mid l_R(e) \in FRAME\}$ such that $R' = R - B$. Moreover, by definition of $FRAME$, for all $e \in B$, we can find a derivation $l_R(e)^* \xRightarrow{*} frame(l_R(e)^*)$ in HRG . Joint embedding of these derivations into R yields a derivation $R \xRightarrow{*} R'$ in HRG . Thus, for the direct derivation $A^* \Rightarrow R''$ through (A, R') in HRG' we can find a derivation $A^* \Rightarrow R \xRightarrow{*} R'$ in HRG . Since $R' \cong R''$, there is a derivation $A^* \Rightarrow R \xRightarrow{*} R''$ in HRG , too. Consider now the remaining derivation $R'' \xRightarrow{k} H$ in HRG' . By construction of HRG' , the new symbol $S' \in N'$ does not appear in R'' . Thus, all hyperedges in R'' are labeled in $N \cup T$. By inductive hypothesis, for each fibre $l_{R''}(e)^* \xRightarrow{k} H(e)$ of $R'' \xRightarrow{k} H$ induced by $e \in E_{R''}$ with $l_{R''}(e) \in N$, there is a derivation $l_{R''}(e)^* \xRightarrow{*} H(e)$ in HRG . Joint embedding of these derivations into R'' yields a derivation $R'' \xRightarrow{*} H$ in HRG (see Context-Freeness Lemma). Moreover, $H \not\cong frame(A^*)$ because $size(frame(A^*)) < size(A^*)$ and all productions of P' are size preserving or

increasing (with the possible exception $(S', \text{frame}(Z))$, which is not applicable because $S' \neq A \in N$ and S' does not appear on the right-hand side of any production). This completes the inductive proof.

Claim 2.2: For each $A \in N$ and each derivation $A^* \xRightarrow{*} H$ in HRG with $H \not\cong \text{frame}(A^*)$ there is a derivation $A^* \xRightarrow{*} H$ in HRG' .

Proof of Claim 2.2 (by induction on the number of steps in the derivation): The statement is trivially true for one-step derivations. Suppose that the statement is true for k -step derivations. Let $A^* \xRightarrow{k+1} H$ be a $(k+1)$ -step derivation in HRG and $H \not\cong \text{frame}(A^*)$. Then the derivation decomposes into $A^* \xRightarrow{*} R'' \xRightarrow{*} H$ for some $R'' \in \mathcal{H}_{N \cup T}$. If the first derivation step is a dummy, then $A^* \cong R''$, and $R'' \xRightarrow{*} H$ can be modified into $A^* \xRightarrow{*} H$. By inductive hypothesis, there is a derivation $A^* \xRightarrow{*} H$ in HRG' . Otherwise, a production $(A, R) \in P$ is applied. Then $R \cong R''$ and $R'' \xRightarrow{*} H$ can be modified into a derivation $R \xRightarrow{*} H$. Consider now the fibres $l_R(e)^* \xRightarrow{*} H(e)$ of $R \xRightarrow{*} H$ for $e \in E_R$. Choosing $B = \{e \in E_R \mid H(e) \cong \text{frame}(l_R(e)^*)\}$, we get $l_R(e) \in \text{FRAME}$ for all $e \in B$. Since $H \not\cong \text{frame}(A^*)$, $R - B \not\cong \text{frame}(A^*)$. Thus, $(A, R - B)$ is a production in P' . By inductive hypothesis, for each fibre $l_R(e)^* \xRightarrow{*} H(e)$ in HRG with $H(e) \not\cong \text{frame}(l_R(e)^*)$, there is a corresponding derivation $l_R(e)^* \xRightarrow{*} H(e)$ in HRG' . Joint embedding of the fibres with $l_R(e) \in E_R - B$ into $R - B$ yields a derivation $R - B \xRightarrow{*} H$ in HRG' . Composing the derivations, we get a derivation $A^* \xRightarrow{*} H$ in HRG' . This completes the inductive step.

Substituting $l(Z)$ for A in Claim 2.1 and Claim 2.2, we see that for $H \not\cong \text{frame}(Z)$, $Z \xRightarrow{*}_{P'} H$ if and only if $Z \xRightarrow{*}_P H$. Now two cases may occur.

Case 1: $l(Z) \notin \text{FRAME}$. Then $Z' = Z$ and, for each $H \in L(HRG)$, $H \not\cong \text{frame}(Z)$. Thus, $H \in L(HRG')$ if and only if $H \in L(HRG)$ is evident.

Case 2: $l(Z) \in \text{FRAME}$. Then $Z' = S'^*$ and there are two productions with left-hand side S' : The production (S', Z) relates the new axiom Z' and the old one, Z , so that for $H \not\cong \text{frame}(Z)$, $Z' \xRightarrow{*}_{P'} H$ if and only if $Z \xRightarrow{*}_P H$. The production $(S', \text{frame}(Z))$ allows the generation of all $H \in \mathcal{H}_T$ with $H \cong \text{frame}(Z)$. $l(Z) \in \text{FRAME}$ means that all $H \in \mathcal{H}_T$ with $H \cong \text{frame}(Z)$ are in $L(HRG)$. Thus, $H \in L(HRG')$ if and only if $H \in L(HRG)$.

Therefore, $L(HRG') = L(HRG)$. □

Typed hyperedge-replacement grammars without empty productions are *monotone* in the following sense: Let $HRG = (N, T, P, Z)$ be a typed hyperedge replacement grammar. Then, for each production $(A, R) \in P$, the *monotony condition*

$$\text{size}(A^*) \leq \text{size}(R)$$

is satisfied with the possible exception of the production $(l(Z), \text{frame}(Z))$ where the label of the axiom, $l(Z)$, does not appear on the right-hand side of any production. This observation has a nice consequence with respect to the Membership Problem for hyperedge-replacement grammars.

1.6 Corollary (Membership Problem)

For hyperedge-replacement grammars, the Membership Problem is decidable, i.e., there is an algorithm for determining whether, given a hyperedge-replacement grammar $HRG = (N, T, P, Z)$ and a multi-pointed hypergraph $H \in \mathcal{H}_T$, H is in $L(HRG)$ or not.

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. By the Typification Theorem (see Chapter I, Theorem 4.3) and Theorem 1.5, we may assume that HRG is a typed grammar without empty productions. Let $H \in \mathcal{H}_T$. Suppose that $\text{size}(H) = k$ for some $k \in \mathbb{N}$. Then — analogously to the string case — the set

$$L(HRG)_k = \{H \in L(HRG) \mid \text{size}(H) \leq k\}$$

can be constructed effectively (up to isomorphic samples).

Then $H \in L(HRG)$ if and only if $H \in L(HRG)_k$. □

Another transformation on hyperedge-replacement grammars — which turns out to be useful — is the removal of productions of the form (A, R) where R is a nonterminal singleton. Productions of this type do not produce any node; they produce a single nonterminal hyperedge. Therefore, they are called *single productions*.

1.7 Definition (single productions)

A production (A, R) over N is said to be a *single production* (with respect to N) if R is a singleton with $l(R) \in N$.

Given a hyperedge-replacement grammar $HRG = (N, T, P, Z)$, we are going to eliminate single productions.

1.8 Theorem (elimination of single productions)

For each hyperedge-replacement grammar, an equivalent grammar without single productions can be constructed.

Remark

Theorem 1.8 is similar to the corresponding theorem for context-free string grammars (see, e.g., [AU 72], Theorem 2.15, [HU 69], Theorem 4.4). However, it should be noticed that the adaptation of the proof from the string case to the hypergraph case becomes possible because of the frequent use of the Context-Freeness Lemma (II.2.4) .

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. By the Typification Theorem (see Chapter I, Theorem 4.3) and Theorem 1.5, we may assume that HRG is a typed grammar without empty productions. We now construct a hyperedge-replacement grammar $HRG' = (N, T, P', Z)$ without single productions using more or less the same methods as in the string case:

(1) Construct the set

$SINGL(A) = \{[H] \mid A^* \xrightarrow{*} H \text{ and } H \text{ is a singleton with } l(H) \in N\}$ for all $A \in N$.³

[This set is finite because N is finite and, for each $B \in N$, there is only a finite number of non-isomorphic singletons with label B .]

(2) Construct P' according to the following rule:

If $[H] \in SINGL(A)$ for some $A \in N$, $l(H) = B$, and $(B, S) \in P$ is not a single production, place (A, S') in P' where S' is a representative of the set $\{H' \mid H \xRightarrow{(B, S)} H'\}$.

[The set P' is finite because P is finite and, for each $A \in N$, $SINGL(A)$ is finite.]

By inspection, $HRG' = (N, T, P', Z)$ is typed and has neither empty nor single productions. Thus, the main tasks of the proof are to show that, for each $A \in N$, the set $SINGL(A)$ can be constructed effectively and that the grammars HRG and HRG' are equivalent.

1. For constructing the set

• $SINGL(A) = \{[H] \mid A^* \xrightarrow{*} H \text{ and } H \text{ is a singleton with } l(H) \in N\}$,

we are going to define sets $SINGL(A)_k$ for $A \in N$ and $k \in \mathbb{N}$ recursively. For a singleton H , $[H]$ is in $SINGL(A)_k$ if there is a derivation $A^* \xrightarrow{j} H$ for some $j \leq k$ and the label of H , $l(H)$, is nonterminal. Note that $A^* \xrightarrow{0} A^*$ for all $A \in N$ so that $[A^*]$ appears already in $SINGL(A)_0$.

• $SINGL(A)_0 = \{[A^*]\}$,

$SINGL(A)_{k+1} = SINGL(A)_k \cup \{[H] \mid [G] \in SINGL(A)_k, (l(G), S) \in P$

• $\text{is a single production, and } G \xRightarrow{(l(G), S)} H\}$

For $A \in N$ and $k \in \mathbb{N}$, $SINGL(A)_k \subseteq SINGL(A)_{k+1} \subseteq SINGL(A)$. Since $SINGL(A)$ is finite, there is some $l \in \mathbb{N}$ such that $SINGL(A)_l = SINGL(A)_{l+1}$. Obviously, all $SINGL(A)_k$ up to the smallest possible l can be constructed effectively. Note that $SINGL(A)_l \subseteq SINGL(A)$. Hence it remains to show that $SINGL(A) \subseteq SINGL(A)_l$.

Claim 1: $SINGL(A) \subseteq SINGL(A)_l$.

Proof of Claim 1: $[H] \in SINGL(A)$ means that there is a derivation $A^* \xrightarrow{*} H$ where H is a singleton with $l(H) \in N$. Thus, we prove the statement

If $A^* \xrightarrow{*} H$ and H is a singleton with label in N , then $[H] \in SINGL(A)_l$.

by induction on the number of derivation steps. If the number of derivation steps is zero, then $H \cong A^*$ so that $[H] = [A^*]$. Hence, $[H] \in SINGL(A)_0 \subseteq SINGL(A)_l$. Assume that the statement holds for all nonterminal singletons which have a derivation of length $j \leq k$.

Consider now a nonterminal singleton H with a derivation $A^* \xrightarrow{k+1} H$ of length $k+1$. Then the derivation decomposes into $A^* \xrightarrow{k} G \xRightarrow{(B, S)} H$ for some $G \in \mathcal{H}_{NUT}$ and $(B, S) \in P$.

Since $[H] \in SINGL(A)$ and since HRG is a hyperedge-replacement grammar without empty productions, $[G] \in SINGL(A)$ and $(l(G), S)$ is a single production. By inductive hypothesis, $[G] \in SINGL(A)_l$. By definition and choice of l , we get

$[H] \in SINGL(A)_{l+1} = SINGL(A)_l$. This completes the inductive proof.

³ For $H \in \mathcal{H}_C$, $[H]$ denotes the set of multi-pointed hypergraphs isomorphic to H , i.e., $[H] = \{H' \in \mathcal{H}_C \mid H' \cong H\}$.

Now $SINGL(A) = SINGL(A)$, which means that the set $SINGL(A)$ can be constructed effectively. Therefore, a hyperedge-replacement grammar $HRG' = (N, T, P', Z)$ can be constructed as outlined above.

It remains to show that $L(HRG') = L(HRG)$. For proving $L(HRG') \subseteq L(HRG)$, we make use of the following observation: For each production $(A, S') \in P'$, there is some R with $[R] \in SINGL(A)$, (possibly $R = A^*$) and some production $(l(R), S) \in P$ such that $A^* \xRightarrow{*} R$ and $R \xRightarrow{(l(R), S)} S'$. Therefore, for each direct derivation $H_i \xRightarrow{(A, S')} H_{i+1}$ in HRG' ,

there is a derivation $H_i \xRightarrow{*} H_{i+1}$ in HRG . If now $Z = H_0 \Rightarrow H_1 \Rightarrow \dots \Rightarrow H_k$ is a derivation in HRG' , then we can find a corresponding derivation $Z = H_0 \xRightarrow{*} H_1 \xRightarrow{*} \dots \xRightarrow{*} H_k$ in HRG . To show $L(HRG) \subseteq L(HRG')$ we claim the following.

Claim 2: For each $A \in N$ and each derivation $A^* \xRightarrow{*} H$ in HRG with $H \in \mathcal{H}_T$, there is a derivation $A^* \xRightarrow{*} H$ in HRG' .

Proof of Claim 2 (by induction on the number of steps in the derivation): The statement is true for one-step derivations. Suppose now that it is true for up to k -step derivations.

Consider a derivation $A^* \xRightarrow{k+1} H$ in HRG' with $H \in \mathcal{H}_T$, say

$A^* = H_0 \Rightarrow H_1 \Rightarrow \dots \Rightarrow H_{k+1} = H$. Since $H \in \mathcal{H}_T$, we can find an index $j \geq 0$ such that $A^* \xRightarrow{j} H_j$ is a derivation through single productions and $H_j \Rightarrow H_{j+1}$ is a direct derivation through a production other than a single production, say (B, S) . Then, $[H_j] \in SINGL(A)$, $l(H_j) = B$, and without loss of generality, (A, H_{j+1}) is a production in P' . $A^* \xRightarrow{k+1} H$ decomposes into the derivations $A^* \xRightarrow{j+1} H_{j+1}$ in HRG' and $H_{j+1} \xRightarrow{k-j} H$ in HRG . Obviously, $H_{j+1} \xRightarrow{k-j} H$ is of length $k - j \leq k$. By inductive hypothesis, for each nonterminal hyperedge e in H_{j+1} and each fibre $l(e)^* \xRightarrow{k-j} H(e)$ of $H_{j+1} \xRightarrow{k-j} H$ there is a derivation $l(e)^* \xRightarrow{*} H(e)$ in HRG' . Joint embedding of these derivations into H_{j+1} yields a derivation $H_{j+1} \xRightarrow{*} H$ in HRG' . [At this place we make use of the Context-Freeness Lemma given in II.2.4.] Composing the derivations $A^* \xRightarrow{j+1} H_{j+1}$, $H_{j+1} \xRightarrow{*} H$ in HRG' , we get a derivation $A^* \xRightarrow{*} H$ in HRG' . This completes the inductive step. $\square$

1.9 Definition (proper and cycle-free grammars)

1. A hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is said to be *proper* if it is typed, well-formed, reduced, and possesses neither empty nor single productions.⁴
2. A typed hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is said to be *cycle-free* if there is no non-trivial derivation⁵ of the form $A^* \xRightarrow{*} S$ where $A \in N$ and S is a singleton with $l(S) = A$.

Now we get the following result as corollary of the theorems above.

⁴ The absence of empty productions is meant in the sense of 1.5.

⁵ A derivation is said to be *non-trivial* if at least one production is applied.

1.10 Corollary

1. For each hyperedge-replacement grammar there is an equivalent proper one.
2. Each proper hyperedge-replacement grammar is cycle-free.

Proof

Statement 1 follows directly from the Typification Theorem, the Well-Formedness Theorem (both given in Chapter I), and the theorems 1.2, 1.5, and 1.8. Statement 2 is an immediate consequence of the fact that for proper grammars the application of a non-terminating production leads to a hypergraph with increased size; the size cannot be reduced again by applying further productions. $\square$

2. A Pumping Lemma for HRL's

The main result presented in this section is a Pumping Lemma for hyperedge-replacement languages. Roughly speaking, it says that each sufficiently large hypergraph belonging to a hyperedge-replacement language can be decomposed into three hypergraphs *FIRST*, *LINK*, and *LAST*, so that a suitable composition of *FIRST*, *LAST*, and k samples of *LINK* for each natural number k yields also a member of the language. This theorem generalizes the well-known Pumping Lemma for context-free string languages — also named Bar-Hillel Lemma, uvwxyz Lemma (cf., e.g., [BPS 61], [HU 69]) — and it is closely related to the Pumping Lemma for context-free graph languages given in [Kr 79].

For a precise formulation of the Pumping Lemma for hyperedge-replacement languages we use an operation which allows to compose hypergraphs in a suitable way.

2.1 Definition (composition and iterated composition)

1. Let $X \in C$ and $m, n \in \mathbb{N}$. Then $H \in \mathcal{H}_C$ is said to be $X(m, n)$ -handled if there is a unique (m, n) -edge $e \in E_H$ with label X , i.e., $l_H(e) = X$. In this case, e is called the $X(m, n)$ -handle of H ; the hypergraph without the $X(m, n)$ -handle, $H - e$, is denoted by H_0 .
2. Let $H \in \mathcal{H}_C$ be a multi-pointed hypergraph¹ with $X(m, n)$ -handle e and let $H' \in \mathcal{H}_C$ be an (m, n) -hypergraph. Then the multi-pointed hypergraph $REPLACE(H, e \mapsto H')$ is called the *composition* of H and H' with respect to e and is abbreviated by $H \otimes H'$.
3. Let $H \in \mathcal{H}_C$ be an (m, n) -hypergraph with some $X(m, n)$ -handle e . Then, for $k \in \mathbb{N}$, $H^k \in \mathcal{H}_C$ is recursively defined as follows:
 - (1) $H^0 = X(m, n)^*$,
 - (2) $H^{i+1} = H \otimes H^i$ for $i \geq 0$.²

Remarks

1. The composition of a multi-pointed hypergraph H with $X(m, n)$ -handle e and an (m, n) -hypergraph H' with respect to e can be constructed by removing e from H and inserting H' instead of it. This can be depicted as follows:



2. Let $H \otimes H'$ be the result of the composition of H and H' with respect to the $X(m, n)$ -handle e in H . Then $H - e$ is a subhypergraph of $H \otimes H'$ (up to isomorphism). Moreover, there is a morphism $embed : H' \rightarrow H \otimes H'$ where $embed_V : V_{H'} \rightarrow V_{H \otimes H'}$ is defined by $embed_V(begin_{H', i}) = s_H(e)_i$, $embed_V(end_{H', j}) = t_H(e)_j$ for $begin_{H', i}, end_{H', j} \in EXT_H$.³

¹ For simplicity, we assume that, for $H \in \mathcal{H}_C$ with $type(H) = (m, n)$, $EXT_H = m + n$.

² For $i \in \mathbb{N}$, H^i is an (m, n) -hypergraph. Thus, $H \otimes H^i$ can be constructed.

³ For $w \in V^*$ and $i \leq |w|$, w_i denotes the i -th symbol of the string w .

and $embed_V(v) = v$ otherwise and $embed_E : E_{H'} \rightarrow E_{H \otimes H'}$ is defined by $embed_E(e) = e$ for all $e \in E_{H'}$.

3. $\otimes$ is an associative operation in the following sense: Let $F \in \mathcal{H}_C$, $G, H \in \mathcal{H}_C$ be (m, n) -hypergraphs, and F and G be $X(m, n)$ -handled. Then $F \otimes G$ is $X(m, n)$ -handled and $(F \otimes G) \otimes H = F \otimes (G \otimes H)$ (up to isomorphism).

4. The (m, n) -hypergraph $X(m, n)^*$ plays a distinguished role with respect to the composition defined above:

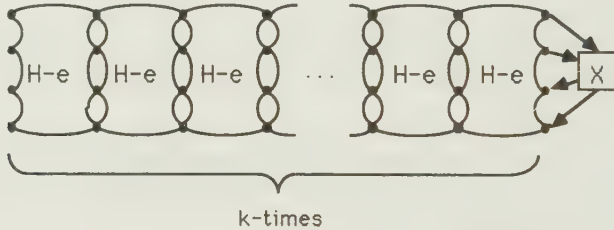
(1) Let H be an (m, n) -hypergraph. Then $X(m, n)^* \otimes H = H$ (up to isomorphism).

(2) Let $H \in \mathcal{H}_C$ be $X(m, n)$ -handled. Then $H \otimes X(m, n)^* = H$ (up to isomorphism).

E.g., this specific hypergraph has the property of being a unit.

5. By construction, $H^1 = H \otimes X(m, n)^* = H$ (up to isomorphism).

6. For $k \in \mathbb{N}$, H^k becomes an (m, n) -hypergraph. Intuitively, H^k is created from k samples of the hypergraph H which are composed with each other. This can be depicted as follows:



One more prerequisite is required for our Pumping Lemma. Hyperedge-replacement grammars can be classified according to the upper bounds of the numbers of external nodes in the right-hand sides of their productions, which will be called *orders*. Actually, we are going to formulate a Pumping Lemma for each order. The resulting classification of hyperedge-replacement languages will be subject of a more detailed study in Section 5.

2.2 Definition (order)

1. A hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is said to be of order r (for some $r \in \mathbb{N}$) if the number of external nodes in the right-hand sides of their productions is bounded by r , i.e., for all $(A, R) \in P$, $|EXT_R| \leq r$. The class of all hyperedge-replacement grammars of order r is denoted by $\mathcal{HRG}_r$.

2. A hyperedge-replacement language L is of order r (for some $r \in \mathbb{N}$) if there is a hyperedge-replacement grammar HRG of order r with $L(HRG) = L$. The class of all hyperedge-replacement languages of order r is denoted by $\mathcal{HRL}_r$; the class of all hyperedge-replacement languages is denoted by $\mathcal{HRL}$.

Remark

Obviously, we have $\mathcal{HRL}_0 \subseteq \mathcal{HRL}_1 \subseteq \mathcal{HRL}_2 \subseteq \mathcal{HRL}_3 \subseteq \dots$. It will be shown that all the inclusions are proper (see Section 5).

2.3 Theorem (Pumping Lemma)

Let L be a hyperedge-replacement language of order r (for some $r \in \mathbb{N}$).

Then there exist constants p and q depending only on L , such that the following is true:

For each multi-pointed hypergraph H in L with $\text{size}(H) > p$ ⁴ there are

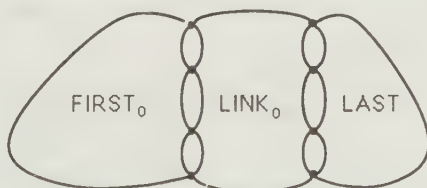
- a symbol $X \in C$,
- natural numbers $m, n \in \mathbb{N}$,
- an $X(m, n)$ -handled multi-pointed hypergraph $FIRST$,
- an $X(m, n)$ -handled (m, n) -hypergraph $LINK$, and
- an (m, n) -hypergraph $LAST$

such that

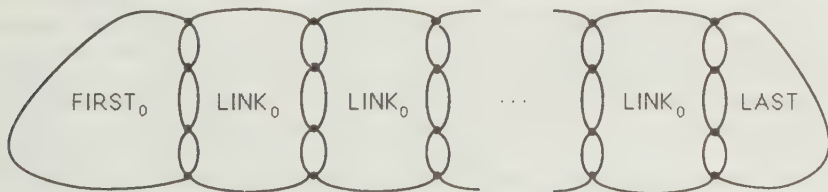
- (1) $H = FIRST \otimes LINK \otimes LAST$,
- (2) $\text{size}(LINK \otimes LAST) \leq q$,
- (3) $\text{size}(\text{frame}(LINK)) \leq r$,
- (4) $LINK$ is non-trivial, i.e., $LINK$ is not a singleton,
- (5) for $k \in \mathbb{N}$, $FIRST \otimes LINK^k \otimes LAST$ is in L .

Remarks

1. The multi-pointed hypergraph H is composed of the hypergraphs $FIRST_0$, $LINK_0$, and $LAST$. This can be depicted by



Since $LINK$ and $LAST$ are of the same type, $LAST$ — instead of $LINK$ — may be composed with $FIRST$. Moreover, $LINK$ — instead of $LAST$ — may be composed with $LINK$. Hence pumping is possible. The pumped hypergraphs have the shape



2. In some sense the pictures above are overly simplifying, because they suggest that $FIRST_0$, $LINK_0$, and $LAST$ are connected hypergraphs — but they have not to be connected in general (see Example 2.4).

⁴ For $H \in \mathcal{H}_C$, $\text{size}(H)$ denotes the number of nodes and hyperedges in H , i.e., $\text{size}(H) = |V_H| + |E_H|$. Moreover, $\text{intsize}(H)$ denotes the number of internal nodes and hyperedges, i.e., $\text{intsize}(H) = |\text{INT}_H| + |E_H|$.

3. Note that the number of external nodes of the *LINK*-component is bounded by r . This requirement restricts the number of possible decompositions of H .
4. Property (4) guarantees that the pumped multi-pointed hypergraphs are different from the given one and from each other.
5. The Pumping Lemma states that for sufficiently "large" hypergraphs "pumping" is possible without leaving the class of considered hypergraphs. In Theorem 2.3, we regard to the *size* of hypergraphs, i.e., the number of nodes plus the number of hyperedges. Similarly, we may consider the *number of nodes* resp. the *number of hyperedges* as relevant counting units. Replacing the condition " $size(H) > p$ " by " $|V_H| > p$ " resp. " $|E_H| > p$ ", we obtain weaker versions of the Pumping Lemma in 2.3.
6. The pumping property is a necessary condition for a hypergraph language to be a hyperedge-replacement language. One of the main uses of the Pumping Lemma is to prove that specific hypergraph languages are not in $\mathcal{HRL}$ resp. not in $\mathcal{HRL}$, (see Sect. 5).

Proof (of the Pumping Lemma)

Let L be a hyperedge-replacement language of order r . By Corollary 1.10, we may assume that L is generated by a proper grammar $HRG = (N, T, P, Z)$ of order r .

[Note that the transformations of grammars presented in Section 1 do not influence the order: If a grammar is of order r , then the transformed grammar is of order r , too.]

Now choose

$$p = \max^n + c \text{ and } q = \max^n + c$$

where c denotes the number of external nodes of Z , $\max$ denotes the maximal number of internal items (internal nodes and hyperedges) of the right-hand sides of productions, i.e., $\max = \max\{\text{intsize}(R) \mid (A, R) \in P\}$, and n denotes the number of nonterminals in HRG .

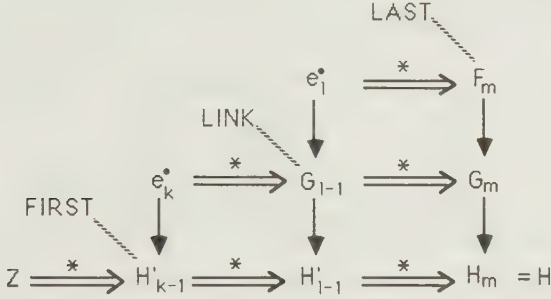
In the following we will show that each multi-pointed hypergraph H in L with $size(H) > p$ can be decomposed into multi-pointed hypergraphs *FIRST*, *LINK*, and *LAST* such that pumping in the sense of the theorem is possible.

Let H be a multi-pointed hypergraph in L with $size(H) > p$ and

der : $Z \Rightarrow H_1 \Rightarrow \dots \Rightarrow H_m = H$ be a derivation of H in HRG . Without loss of generality, we may assume that *der* is a canonical derivation, i.e., in each derivation step all nonterminal hyperedges are replaced and there are no dummy derivations. Then we can find a sequence of productions $p_1, \dots, p_m$ such that, for each $i = 1, \dots, m-1$, p_i and p_{i+1} are dependent in *der*, i.e., the nonterminal hyperedge replaced by p_{i+1} is just generated by p_i in the preceding direct derivation of *der*. The left-hand sides of those productions form a sequence of nonterminals $A_1, \dots, A_m$. If the length m of the derivation *der* is greater than the number n of nonterminals in N (and this is guaranteed by $size(H) > \max^n + c$, see Claim 2 below), at least two items in this sequence must be equal. Then choose $A_k = A_l$ with $1 \leq k < l \leq m$ and $m - k \leq n$. Consequently, one nonterminal $A = A_l$ occurs in a multi-pointed hypergraph G_{l-1} derived from e_k^* with $l(e_k) = A$. Such a situation provides a self-embedding property (c.f. [HU 69], 4.5).

We will decompose the derivation of H in a suitable way. Let us mention that the canonical derivation $Z \Rightarrow H_{k-1} \Rightarrow H_{l-1} \Rightarrow H_m$ can be transformed into an equivalent derivation $Z \Rightarrow H'_{k-1} \Rightarrow H'_{l-1} \Rightarrow H_m$ where H'_{k-1} and H'_{l-1} are terminal up to one

nonterminal hyperedge with label A . Let now $e_k^* \xRightarrow{*} G_{i-1} \xRightarrow{*} G_m$ be the restriction of $H_{k-1}^i \xRightarrow{*} H_{i-1}^* \xRightarrow{*} H_m$ to A^* . Then G_{i-1} is terminal up to one nonterminal hyperedge with label A . Finally, let $e_i^* \xRightarrow{*} F_m$ be the restriction of the derivation $G_{i-1} \xRightarrow{*} G_m$ to A^* . This situation may be depicted as follows:



Then, by the Context-Freeness Lemma, $G_m = \text{REPLACE}(G_{i-1}, e_i \mapsto F_m)$ and $H_m = \text{REPLACE}(H'_{k-1}, e_k \mapsto G_m)$ where e_i and e_k denote the nonterminal hyperedges in G_{i-1} and H'_{k-1} , respectively. Choosing now $\text{FIRST} = H'_{k-1}$, $\text{LINK} = G_{i-1}$, and $\text{LAST} = F_m$ and using the notation introduced in Definition 2.1, we get $H = \text{FIRST} \otimes (\text{LINK} \otimes \text{LAST})$.

By construction, $\text{type}(\text{FIRST}) = \text{type}(H)$. Since the considered hyperedge-replacement grammar is typed and the hyperedges e_k and e_i possess the same label, e_k and e_i are hyperedges of the same type, say of type $(\hat{m}, \hat{n})$. Thus, LINK and LAST are of the same type, namely of type $(\hat{m}, \hat{n})$. Since LINK is an $A(\hat{m}, \hat{n})$ -handled $(\hat{m}, \hat{n})$ -hypergraph, for each $i \in \mathbb{N}$, LINK' is defined and turns out to be $A(\hat{m}, \hat{n})$ -handled. Since FIRST is $A(\hat{m}, \hat{n})$ -handled and LAST is a $(\hat{m}, \hat{n})$ -hypergraph, for $i \in \mathbb{N}$, $\text{FIRST} \otimes (\text{LINK}' \otimes \text{LAST})$ can be constructed. On the other hand, for $i \in \mathbb{N}$, $\text{FIRST} \otimes (\text{LINK}' \otimes \text{LAST})$ can be derived from the axiom Z : We use the derivation $Z \xRightarrow{*} \text{FIRST}$, a derivation $e_k^* \xRightarrow{*} \text{LINK}'$ (composed by the derivation $e_k^* \xRightarrow{*} \text{LINK}$), and the derivation $e_i^* \xRightarrow{*} \text{LAST}$.

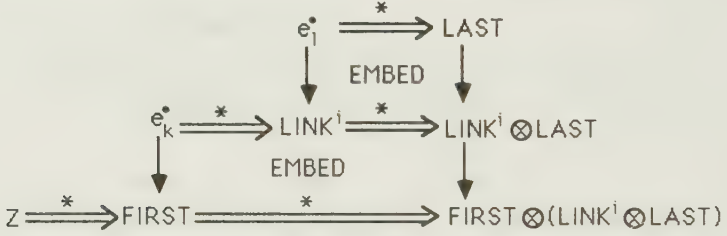
Claim 1: For each $i \in \mathbb{N}$, there is a derivation $e_k^* \xRightarrow{*} \text{LINK}'^i$.

Proof of Claim 1 (by induction on i): By definition,

$\text{LINK}^0 = A(\hat{m}, \hat{n})^*$, i.e., $\text{LINK}^0 \cong e_k^*$. Therefore, there is a derivation $e_k^* \xRightarrow{0} \text{LINK}^0$. Suppose that the statement is true for some $i \geq 0$. Now $\text{LINK}'^{i+1} = \text{LINK} \otimes \text{LINK}'^i$. By choice of LINK , there is a derivation $e_k^* \xRightarrow{*} \text{LINK}$ such that LINK is $A(\hat{m}, \hat{n})$ -handled (with handle e_i). By inductive hypothesis, there is a derivation $e_k^* \xRightarrow{*} \text{LINK}'^i$. The embedding of this derivation into LINK yields the derivation $\text{LINK} \xRightarrow{*} H^{(i)}$ with $H^{(i)} = \text{REPLACE}(\text{LINK}, e_k \mapsto \text{LINK}') = \text{LINK} \otimes \text{LINK}' = \text{LINK}'^{i+1}$. Therefore, we get a derivation $e_k^* \xRightarrow{*} \text{LINK} \xRightarrow{*} \text{LINK}'^{i+1}$. This completes the inductive proof.

Now, for each $i \in \mathbb{N}$, a derivation $Z \xRightarrow{*} FIRST \otimes (LINK^i \otimes LAST)$ can be constructed according to the following rules:

- (1) Embed the derivation $e_i^* \xRightarrow{*} LAST$ into $LINK^i$.
This leads to a derivation $LINK^i \xRightarrow{*} LINK^i \otimes LAST$.
- (2) Embed the derivation $e_k^* \xRightarrow{*} LINK^i \xRightarrow{*} LINK^i \otimes LAST$ into $FIRST$.
This yields a derivation $FIRST \xRightarrow{*} FIRST \otimes (LINK^i \otimes LAST)$.
- (3) Compose the obtained derivation with the derivation $Z \xRightarrow{*} FIRST$.



Hence, for each integer $i \in \mathbb{N}$, $FIRST \otimes (LINK^i \otimes LAST)$ belongs to the language L generated by the hyperedge-replacement grammar HRG .

To complete the proof, we have to show Claim 2 (see below) and to verify the additional requirements.

Claim 2: If H is in L and $size(H) > \max^n + c$, then the length of each derivation of H is greater than n .

Proof of Claim 2: Consider a derivation of H of length m . Say $H_{m-1} \Rightarrow H_m = H$ is the last derivation step. Then at most all hyperedges of H_{m-1} are replaced simultaneously by at most the maximal number $\max^5$ of internal items of right-hand sides of productions: This implies

$$\text{intsize}(H_m) \leq |INT_{H_{m-1}}| + \max \cdot |E_{H_{m-1}}| \leq \max \cdot \text{intsize}(H_m).$$

Inductively one obtains

$$\text{intsize}(H) \leq \max^m \cdot \text{intsize}(Z) = \max^m.$$

Thus, $size(H) \leq \max^m + size(frame(Z)) = \max^m + c$. Now $size(H) > \max^n + c$ implies $m > n$.

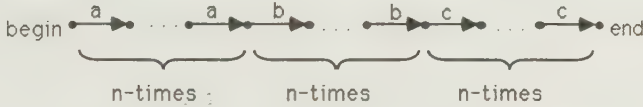
By Claim 2, the length m of the canonical derivation — considered at the beginning — is greater than the number n of nonterminals in the grammar. By choice of k , the derivation $e_k^* \xRightarrow{*} LINK \otimes LAST$ is of length $m - k \leq n$. Using the relation mentioned in Claim 2, one obtains $size(LINK \otimes LAST) \leq \max^n + c = q$. Moreover, $size(frame(LINK)) \leq r$ because HRG is a grammar of order r . Finally, $LINK$ is nontrivial because HRG is a grammar without single productions (and thus cycle-free). This completes the proof. $\square$

⁵ If H is in L and $size(H) > p$, then there is a production (A, R) in P with $\text{intsize}(R) \geq 1$, i.e., $\max \geq 1$: Suppose $\text{intsize}(R) = 0$ for each production (A, R) in P , i.e., $\max = 0$. Then, for each derivation $Z \xRightarrow{*} H \in \mathcal{H}_T$, it would follow that $size(H) = size(frame(Z)) = \max^n + c = p$ which is a contradiction to $size(H) > p$.

We continue this section by the discussion of an example illustrating the pumping property.

2.4 Example (non-context-free string languages)

As shown in Chapter I, the string-graph language $L = \{(a^n b^n c^n)^* \mid n \geq 1\}$ can be generated by a hyperedge-replacement grammar of order 4, i.e., $L = \{(a^n b^n c^n)^* \mid n \geq 1\}$ is a hyperedge-replacement language of order 4. It possesses the pumping property. This may be seen as follows: A string graph $(a^n b^n c^n)^*$ is a (1,1)-hypergraph of the following form:



For $n \geq 3$ the string graph $(a^n b^n c^n)^*$ can be decomposed as indicated below.

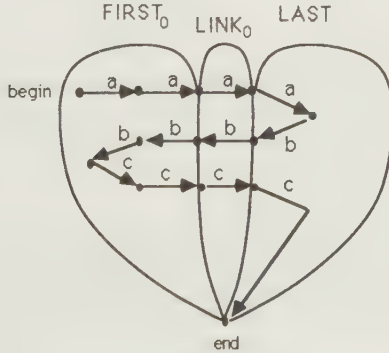


Figure 2.1: Decomposition of the string graph $(a^4 b^4 c^4)^*$.

FIRST is a (1,1)-hypergraph containing the distinguished nodes, *begin* and *end*, of the string graph as distinguished nodes. It consists of 3 chains of edges of length $n - 2$, an *a*-chain, a *b*-chain, and a *c*-chain, where the *a*-chain is attached to the distinguished *begin*-node and the *c*-chain is attached to the end of the *b*-chain. Moreover, it may be seen as (2,2)-handled, where the first pair of tentacles grips to the end of the *a* chain and the begin of the *b*-chain and the second pair of tentacles grips to the end of the *c*-chain and the distinguished *end*-node. It is composed with the *LINK*-component with respect to the (2,2)-handle. The *LINK*-component consists of an *a*-edge, a *b*-edge, and a *c*-edge. Moreover, it possesses a (2,2)-handle which grips to the target of the *a*-edge, the source of the *b*-edge, the target of the *c*-edge and the distinguished *end*-node. Finally, the *LAST*-component consists of an *a*-edge, a *b*-edge, and a *c*-edge where the *b*-edge is attached to the *a*-edge and the *c*-edge ends in the distinguished *end*-node. Note, that

neither $FIRST_0$ ($FIRST$ without the (2,2)-handle) nor $LINK_0$ ($LINK$ without the (2,2)-handle) nor $LAST$ is connected. Moreover, $LINK$ is non-trivial. Now the pumped (1,1)-hypergraphs have the following shape:

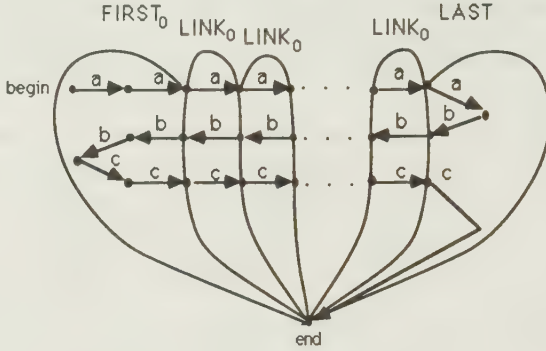


Figure 2.2: A pumped (1,1)-hypergraph.

Obviously, each resulting (1,1)-hypergraph is a string graph of the string-graph language $L = \{(a^n b^n c^n)^* | n \geq 1\}$.

2.5 Corollary (Finiteness Problem)

For hyperedge-replacement grammars, the Finiteness Problem is decidable, i.e., there is an algorithm to determine whether a given hyperedge-replacement grammar generates a finite or infinite number of non-isomorphic hypergraphs.

Proof

Let HRG be a proper hyperedge-replacement grammar and p and q be the constants defined in the proof of Theorem 2.3.

If H is in $L(HRG)$ and $\text{size}(H) > p$, then H can be written as $H = FIRST \otimes LINK \otimes LAST$ such that, for each $k \geq 0$, $FIRST \otimes LINK^k \otimes LAST$ is in $L(HRG)$. Also $LINK$ is non-trivial, i.e., $\text{intsize}(LINK) > 1$. This implies that for each $k \geq 0$,

$$\text{size}(FIRST \otimes LINK^k \otimes LAST) < \text{size}(FIRST \otimes LINK^{k+1} \otimes LAST).$$

Hence, if there is a hypergraph H in $L(HRG)$ with $\text{size}(H) > p$, then the number of non-isomorphic hypergraphs in $L(HRG)$ is infinite.

Conversely, if the number of non-isomorphic hypergraphs in $L(HRG)$ is infinite, then there are arbitrary "large" hypergraphs in $L(HRG)$; in particular, there is a hypergraph $H \in L(HRG)$ with $\text{size}(H) > p + q$. This hypergraph can be written as

$FIRST \otimes LINK \otimes LAST$ where $1 \leq \text{size}(LINK \otimes LAST) \leq q$, $\text{intsize}(LINK) > 1$, and, for all $k \geq 0$, $FIRST \otimes LINK^k \otimes LAST$ is in $L(HRG)$. In particular, $H^{(0)} = FIRST \otimes LINK^0 \otimes LAST (\cong FIRST \otimes LAST)$ is in $L(HRG)$ and $\text{size}(H^{(0)}) < \text{size}(H)$.

Also $\text{size}(H^{(0)}) > p$. If $\text{size}(H^{(0)}) > p + q$, we repeat the procedure until we eventually find a hypergraph G in L with $p < \text{size}(G) \leq p + q$. Thus we have:

$L(HRG)$ contains an infinite number of non-isomorphic hypergraphs
if and only if $L(HRG)$ contains a hypergraph G with $p < \text{size}(G) \leq p + q$.

Since we may test whether a given hypergraph is a member in a given hyperedge-replacement language (Corollary 1.6), we have merely to test all (non-isomorphic) hypergraphs of size between p and $p + q$ for membership in $L = L(HRG)$. If there is such a hypergraph, then L contains clearly an infinite number of non-isomorphic hypergraphs; if not, then there are no hypergraphs of size greater than p in L , so that the number of non-isomorphic hypergraphs is finite. $\square$

One of the main uses of results like the Pumping Lemma 2.3 is to prove that specific hypergraph languages are not hyperedge-replacement languages — a task which is rather difficult in general. We derive now a theorem from the Pumping Lemma which applies very nicely to show that specific hypergraph languages cannot be generated by hyperedge-replacement grammars.

2.6 Theorem (sublinear growing)

Let L be an infinite hyperedge-replacement language. Then there exist an infinite sequence of hypergraphs in L , say $H_0, H_1, H_2, \dots$, and constants $c, d \in \mathbb{N}$ with $c + d > 0$ such that for all $i \geq 0$

$$|V_{H_{i+1}}| = |V_{H_i}| + c \quad \text{and} \quad |E_{H_{i+1}}| = |E_{H_i}| + d.$$

Proof

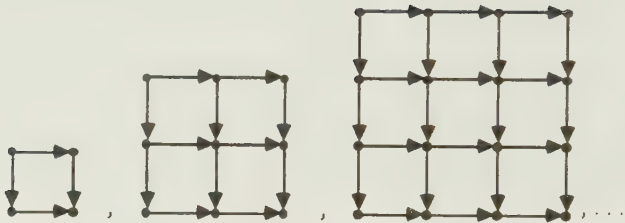
The desired sequence of hypergraphs can be obtained by pumping up a sufficiently large hypergraph H_0 in L . The constant c can be chosen to be the number of internal nodes of $LINK$ and the constant d can be chosen as the number of hyperedges in $LINK$ minus 1. Because $LINK$ is not a singleton, we get $c \neq 0$ or $d \neq 0$. $\square$

2.7 Example (complete graphs)

The set of all complete graphs cannot be generated by a hyperedge-replacement grammar because their growing with respect to the number of hyperedges is not sublinear as required by Corollary 2.6.

2.8 Example (square grids)

The set of all square grids of the form



cannot be generated by a hyperedge-replacement grammar because they grow too fast.

3. Parikh's Theorem for HRL's

We now present a result, called Parikh's Theorem, which connects hyperedge-replacement languages with semilinear sets and forms an adaption of Parikh's Theorem for context-free string languages to hyperedge-replacement languages.

Parikh's Theorem for context-free string languages was published originally by Parikh in [Pa 61] and was republished in [Pa 66] due to its importance and its inaccessibility of the original note. It states that counting the occurrences of symbols in words of a context-free language yields a semilinear set. The proof given in [Pa 66] involves a complicated rearrangement of derivation trees. Essentially the same proof may be found in the books of Ginsburg [Gi 66] and Salomaa [Sa 73]. In 1972, Greibach has given still another proof, using a Kleene-type characterization of context-free languages due to Gruska [Gr 71] in terms of an iterated substitution operator. In [Go 77], Goldstine has shown that a slightly strengthened form of the classical Pumping Lemma for context-free languages can be used to give a simplified proof of Parikh's Theorem. It should be stressed that his proof is conceptually the same as Parikh's original proof; it is somewhat clearer because it makes use of a generalized version of the Pumping Lemma instead of manipulating derivations resp. derivation trees.

In the following we adapt Parikh's Theorem to hyperedge-replacement languages. Parikh's, Greibach's as well as Goldstine's proofs are candidates for adaption: Derivation trees are introduced in II.3, a Kleene-type characterization for hyperedge-replacement languages is presented in III.2, and a Pumping Lemma is formulated and proved in the previous section. We present a proof based on derivation trees.

3.1 Definition (Parikh mapping)

1. Let $T = \{a_1, \dots, a_n\}$ be an alphabet and $\psi : \mathcal{H}_T \rightarrow \mathbb{N}^n$ be the mapping given by

$$\psi(H) = (\#_{a_1}(H), \dots, \#_{a_n}(H)),$$

where $\#_{a_i}(H)$ denotes the number of a_i -labeled hyperedges in H . Then ψ is termed a *Parikh mapping*.¹

2. For any language $L \subseteq \mathcal{H}_T$, $\psi(L)$ denotes the set

$$\psi(L) = \{\psi(H) | H \in L\}.$$

¹ For each integer $n \geq 1$, $\mathbb{N}^n = \mathbb{N} \times \dots \times \mathbb{N}$ denotes the cartesian product of $\mathbb{N}$ with itself n times. For elements $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ in $\mathbb{N}^n$ and c in $\mathbb{N}$, $x + y = (x_1 + y_1, \dots, x_n + y_n)$, $x - y = (x_1 - y_1, \dots, x_n - y_n)$, and $cx = (cx_1, \dots, cx_n)$.

3.2 Definition (linear and semilinear sets)

1. A set $S \subseteq \mathbb{N}^n$ is *linear* if S is of the form

$$S = \left\{ x_0 + \sum_{i=1}^k c_i x_i \mid c_1, \dots, c_k \in \mathbb{N} \right\},$$

where $k \geq 1$ and $x_1, \dots, x_k \in \mathbb{N}^n$.

2. A set $S \subseteq \mathbb{N}^n$ is *semilinear* if it is the finite union of linear sets.

The fundamental result concerning the distribution of labeled hyperedges in hypergraphs is presented in the following theorem.

3.3 Theorem (Parikh's Theorem)

For any hyperedge-replacement language L , the set $\psi(L)$ is semilinear.

Proof

Let L be a hyperedge-replacement language and $HRG = (N, T, P, Z)$ be a generating hyperedge-replacement grammar. Clearly, if $\psi(L - [\text{frame}(Z)])$ is semilinear, then also $\psi(L)$ is semilinear. Hence, without loss of generality, we assume that $[\text{frame}(Z)] \not\subseteq L$. By Corollary 1.10, we may assume that HRG is proper.

For each subset M of N containing $l(Z)$, we consider the subset L_M of L consisting of all hypergraphs which possess at least one derivation from Z where the occurring nonterminals are exactly the nonterminals in the set M . Obviously, L is the union of the languages L_M :

$$L = \bigcup_{\{l(Z)\} \subseteq M \subseteq N} L_M.$$

Since there is only a finite number of M 's, it suffices to show that each of the sets $\psi(L_M)$ is semilinear.

From now on we consider an arbitrary but fixed language L_M . Assume that the set M contains m elements. In what follows, we investigate derivation trees associated with derivations according to HRG .

Let L_1 be the set consisting of all hypergraphs $H \in \mathcal{H}_T$ which are generated by trees such that the following conditions are satisfied.

- (i) The root is labeled by $l(Z)$.
 - (ii) The nodes other than leaves are labeled by elements of M and each element of M occurs at least once as label.
 - (iii) There is no path containing more than $m + 1$ nodes with the same label.
- Clearly, L_1 is a finite subset of L_M (up to isomorphism). We define

$$X = \{\psi(H) \mid H \in L_1\}.$$

For each $A \in M$, let L_A be the set of hypergraphs $H \in \mathcal{H}_{T \cup \{A\}}$ which contain exactly one hyperedge with label A and which are generated by trees such that

- (i)' The root is labeled by A .
- (ii)' The nodes other than leaves are labeled by elements of M .
- (iii)' There is no path containing more than $m + 1$ nodes with the same label.

Clearly, because of condition (iii), L_A is a finite subset of L_M (up to isomorphism). For $H \in L_A$, we denote by $\gamma(H)$ the hypergraph obtained from H by erasing the hyperedge with label A . We now define

$$Y = \{\psi(\gamma(H)) \mid H \in L_A \text{ for some } A \in M\}.$$

Thus, both X and Y are finite sets. Assume that

$$X = \{x_1, \dots, x_r\} \text{ and } Y = \{y_1, \dots, y_s\}.$$

For $i = 1, \dots, r$, let S_i be the set

$$S_i = \left\{ x_i + \sum_{j=1}^s c_j y_j \mid c_1 \dots c_s \in \mathbb{N} \right\}$$

and let S be the union of the S_i . By definition, S is semilinear. To prove Theorem 3.3, it suffices to show that

$$\psi(L_M) = S.$$

We will prove that each side is included in the other.

Assume that $z \in S$. If z is one of the vectors x_i , then by definition of the set X , $z \in \psi(L_1)$, and consequently $z \in \psi(L_M)$.

Proceeding inductively, we assume that $z_1 \in S \cap \psi(L_M)$ and claim that $z = z_1 + y_j \in \psi(L_M)$ where j , $1 \leq j \leq s$, is arbitrary.

By definition of Y , there exist a symbol $A \in M$ and a hypergraph $H \in L_A$ such that $y_j = \psi(\gamma(H))$. Since $H \in L_A$, there is a derivation tree T for H whose root is labeled by A . Furthermore, H contains exactly one hyperedge with label A , and H contains no other nonterminal-labeled hyperedges. The nonterminals occurring as node labels in T belong to M .

By $z_1 \in S \cap \psi(L_M)$ and the definition of L_M , there is a hypergraph $H_1 \in L$ with $z_1 = \psi(H_1)$ and a derivation $Z \xRightarrow{*} H_1$ such that the associated derivation tree T_1 has exactly the elements of M as labels of the nodes other than the leaves. In particular, there is a node in T_1 labeled by A . (In fact, we introduced the language L_M only to be able to make this conclusion.) This node is replaced by the tree T . The resulting derivation tree T_2 gives a hypergraph $H_2 \in L_M$ such that

$$\psi(H_2) = \psi(H_1) + \psi(\gamma(H)) = z_1 + y_j = z.$$

Consequently, for $j = 1, \dots, s$, $z = z_1 + y_j \in \psi(L_M)$ holds true and we have completed the induction. Hence, $S \subseteq \psi(L_M)$.

Conversely, assume that $z \in \psi(L_M)$. Then there is a hypergraph $H \in L_M$ such that $z = \psi(H)$. Let T be the derivation tree associated with the derivation $Z \xRightarrow{*} H$. By definition of L_M , we assume that exactly the elements of M occur as labels of the nodes (other than the leaves) of T . If there is no path in T containing more than $m + 1$ nodes with the same label, then $H \in L_1$. This implies that $z \in X$ and hence $z \in S$.

Thus, we assume that there is a path in T containing at least $m + 2$ nodes with the same label A . We choose $m + 2$ nodes $v_0, \dots, v_{m+1}$ with the same label A in such a way that

- (i) for each $i = 0, \dots, m$, v_{i+1} is in the tree with root v_i and
- (ii) there is no path containing more than $m + 1$ nodes with the same label in the tree T_{v_1} with root v_1 .

(Such nodes v_i are found by checking through T beginning from the leaves.)

There is a natural number l , $1 \leq l \leq m$, such that the same nonterminals occur as node labels in the trees T_{v_l} and $T_{v_{l+1}}$ with roots v_l and v_{l+1} . This follows because there are at most m nonterminals as node labels in the tree T_{v_1} , and at least one nonterminal as node label in the tree $T_{v_{m+1}}$.

We now form a new tree T' by replacing in T the tree T_{v_l} by the tree $T_{v_{l+1}}$. Denote T^* the tree obtained from T_{v_l} by replacing the tree $T_{v_{l+1}}$ by the single node v_{l+1} .

There is no path in T^* containing more than $m + 1$ nodes with the same label A , since there is no such path in the tree T_{v_l} . The hypergraph generated by T^* belongs to the set L_A (by definition of this set). Consequently, if H' is the hypergraph generated by T' , we obtain

$$z = \psi(H) = \psi(H') + y_{j'}$$

where $1 \leq j' \leq s$. Continuing in this fashion, we finally obtain a tree, where there is no path containing more than $m + 1$ nodes with the same label. Thus we can write z in the form

$$z = x_i + \sum_{j=1}^s c_j y_j \quad 1 \leq i \leq r, \quad c_1, \dots, c_s \in \mathbb{N}.$$

But this means that $z \in S$, and, consequently, $\psi(L_M) \subseteq S$. □

Remarks

1. For proving Parikh's Theorem for hyperedge-replacement languages, one may use the Kleene-Type Characterization Theorem presented in Chapter III (2.3). Obviously, (1) finiteness of a language L implies semilinearity of $\psi(L)$. If we can show that (2) semilinearity of $\psi(K)$ and $\psi(L)$ implies semilinearity of $\psi(SUB_A(K, L))$ and (3) semilinearity of $\psi(L)$ implies semilinearity of $\psi(L^A)$, then the Characterization Theorem implies that $\psi(L)$ is semilinear for each hyperedge-replacement language.

2. Beside the given proof, there is a shorter one which directly makes use of Parikh's Theorem for context-free string grammars. It is based on the following idea: Given a proper hyperedge-replacement grammar HRG generating L , construct a context-free string grammar $HRG_{s, \tau}$ with $\psi(L(HRG)) = \psi_{s, \tau}(L(HRG_{s, \tau}))$ where $\psi_{s, \tau}$ denotes the usual Parikh mapping for string languages. By Parikh's Theorem for context-free string languages, $\psi_{s, \tau}(L(HRG_{s, \tau}))$ is semilinear. Consequently, $\psi(L(HRG)) = \psi(L)$ is semilinear.

[Let $\text{str} : \mathcal{H}_{N \cup T} \rightarrow (N \cup T)^*$ be a mapping assigning a string $\text{str}(H) = l_H(e_1) \dots l_H(e_n)$ to a hypergraph H with hyperedges $e_1, \dots, e_n$. Then $HRG_{\text{str}} = (N, T, \text{str}(P), \text{str}(Z))$ with $\text{str}(P) = \{(A, \text{str}(R)) \mid (A, R) \in P\}$ has the desired properties.]

As counting the number of hyperedges (with suitable label), counting the number of nodes in hypergraphs of a hyperedge-replacement language yields a semilinear set.

3.4 Definition (modified Parikh mapping)

Let $T = \{a_1, \dots, a_n\}$ be an alphabet and $\psi' : \mathcal{H}_T \rightarrow \mathbb{N}^{n+1}$ be the mapping given by

$$\psi'(H) = (|V_H|, \#_{a_1}(H), \dots, \#_{a_n}(H)),$$

where $\#_{a_i}(H)$ denotes the number of a_i -labeled hyperedges in H . Then ψ' is termed a *modified Parikh mapping*. Accordingly, $\psi'(L)$ denotes the set $\psi'(L) = \{\psi'(H) \mid H \in L\}$.

3.5 Corollary

For any hyperedge-replacement language L , the set $\psi'(L)$ is semilinear.

Proof

Let L be a hyperedge-replacement language. Then we consider a modified language $L' = \{H' \mid H \in L\}$ where, for $H \in L$, H' is obtained from H by attaching a $(0,1)$ -edge with a label nod (not in T) to each node in H . Obviously, L' is a hyperedge-replacement language, too (attach hyperedges to all nodes of the axiom and to the internal nodes of the right-hand sides). By Theorem 3.3, $\psi(L')$ is semilinear. On the other hand, for $H' \in L'$, $\psi(H') = (\#_{\text{nod}}(H), \#_{a_1}(H), \dots, \#_{a_n}(H)) = (|V_H|, \#_{a_1}(H), \dots, \#_{a_n}(H)) = \psi'(H)$. Therefore, $\psi'(L)$ is semilinear. $\square$

Remarks

1. There is a close relationship to Theorem 2.6 (concerning the sublinear growing) given in the previous section.
2. There are several languages L , e.g., the set of all graphs and the set of all snails (investigated in Section 4) which have the semilinearity property (i.e., $\psi(L)$ is semilinear) but are not hyperedge-replacement languages.

4. Boundedness Properties of HRL's

In this section, we present some consequences of the Pumping Lemma for hyperedge-replacement languages concerning the connectivity, clique size, and minimum degree of its (hyper)graphs. It turns out that, for several graph-theoretic functions (like the connectivity number, the maximum clique size, and the chromatic number), the function values are bounded for members of a given hyperedge-replacement language.

4.1 Definition (simplicity, connectivity)

1. A hypergraph H is *simple* if all $e \in E_H$ are well-formed¹ and there are no multiple hyperedges with same label, i.e., for all $e, e' \in E_H$, $s_H(e) = s_H(e')$, $t_H(e) = t_H(e')$, and $l_H(e) = l_H(e')$ implies $e = e'$.
2. A connected hypergraph H is of *connectivity* k (for some $k \geq 1$), if k is the smallest number of nodes whose removal disconnects H into non-empty components or reduces H to a single node. If H is disconnected, the connectivity of H is defined to be 0. We use $\text{conn}(H)$ to denote the connectivity of H .²

4.2 Theorem (Connectivity Theorem I)

Let L be a hyperedge-replacement language of order r the members of which are simple. Then there exists a constant p depending only on L such that, for each $H \in L$ with $|V_H| > p$, $\text{conn}(H) \leq r$.

Proof

Let L be a hyperedge-replacement language of simple hypergraphs. There there exists a constant r such that L is of order r . Let p and q be the constants occurring in the Pumping Lemma. Then each hypergraph $H \in L$ with $\text{size}(H) > p$ can be written as $\text{FIRST} \otimes \text{LINK} \otimes \text{LAST}$ such that $\text{size}(\text{LINK} \otimes \text{LAST}) \leq q$, $\text{size}(\text{frame}(\text{LINK})) \leq r$, LINK is non-trivial, and, for $k \in \mathbb{N}$, $\text{FIRST} \otimes \text{LINK}^k \otimes \text{LAST}$ is in L . It turns out that LINK possesses at least one internal node. [If LINK has no internal nodes, then it must have a terminal hyperedge connecting external nodes. In this case, there exists an n such that the pumped hypergraph $\text{FIRST} \otimes \text{LINK}^n \otimes \text{LAST}$ has multiple or not well-formed hyperedges, a contradiction.] Removal of the nodes "connecting" the FIRST - and the $\text{LINK} \otimes \text{LAST}$ -part disconnects H into a remainder of FIRST (without handle) and a remainder of $\text{LINK} \otimes \text{LAST}$. Since $\text{size}(H) > p \geq q$, $\text{size}(\text{LINK} \otimes \text{LAST}) \leq q$, and LINK possesses an internal node, the remainders of FIRST and $\text{LINK} \otimes \text{LAST}$ are non-empty. Since $\text{size}(\text{frame}(\text{LINK})) \leq r$, the connectivity of H is bounded by r . $\square$

¹ The notion "well-formed" is meant in the sense of Definition 4.5 in Chapter I.

² Let $H \in \mathcal{H}_C$ and $V \subseteq V_H$. Without loss of generality, we may assume that H is a $(0,0)$ -hypergraph, i.e., $\text{begin}_H = \text{end}_H = \lambda$. Now $H - V$ denotes the hypergraph $(V_H - V, E, s, t, l, \lambda, \lambda)$ where $E = \{e \in E_H \mid s_H(e), t_H(e) \in (V_H - V)^*\}$, and s, t , and l are the restrictions of s_H, t_H , and l_H to E , respectively.

Remarks

1. Connectivity Theorem I can be used to show that the set of all complete graphs cannot be generated by a hyperedge-replacement grammar.
2. For each hyperedge-replacement language L , the language L_{simple} of the underlying simple hypergraphs is a hyperedge-replacement language (see Vogler [Vo 88]). Since for each hypergraph H , $\text{conn}(H) = \text{conn}(H_{\text{simple}})$, Connectivity Theorem I may be formulated for arbitrary hyperedge-replacement languages. [For a hypergraph H , the simple hypergraph H_{simple} underlying H is obtained from H by (1) replacing not well-formed hyperedges by the corresponding well-formed ones and (2) choosing the classes of multiple hyperedges as new hyperedges.]

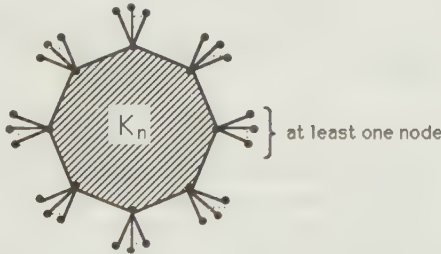
4.3 Corollary (bounded connectivity)

Each hyperedge-replacement language (of simple hypergraphs) is of bounded connectivity.

Proof

Let L be a hyperedge-replacement language of order r (the members of which are simple). Let p be the constant defined in the proof of the Connectivity Theorem I (resp. in the Pumping Lemma). Then, for $H \in L$ with $|V_H| > p$, $\text{conn}(H) \leq r$ and, for $H \in L$ with $|V_H| \leq p$, $\text{conn}(H) \leq p - 1$. Choosing the maximum of r and $p - 1$, we get a bound for all hypergraphs in L . $\square$

The set of all “decorated” complete graphs of the form



(where K_n denotes a complete graph with n nodes) cannot be generated by a hyperedge-replacement grammar although the connectivity is bounded. For proving results of this kind, it is useful to have an additional criterion concerning not only the connectivity of the (hyper)graphs, but also the connectivity of their sub(hyper)graphs.

4.4 Theorem (Connectivity Theorem II)

Let L be a hyperedge-replacement language of simple hypergraphs. Then there exists a natural number k such that, for all $H \in L$, $\text{conn}_{\text{sub}}(H) \leq k$ where $\text{conn}_{\text{sub}}(H) = \max\{\text{conn}(H_{\text{sub}}) | H_{\text{sub}} \subseteq H\}$.

Proof

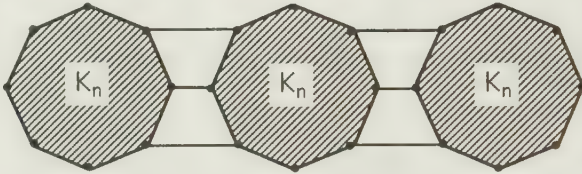
If L is a hyperedge-replacement language, then the set L_{sub} given by

$$L_{\text{sub}} = \{H_{\text{sub}} | H_{\text{sub}} \subseteq H \text{ with } \text{begin}_{H_{\text{sub}}} = \text{begin}_{H_{\text{sub}}} = \lambda\}$$

is a hyperedge-replacement language, too. By Corollary 4.3, L_{sub} is of bounded connectivity, i.e., there is a natural number k such that, for all $G \in L_{\text{sub}}$, $\text{conn}(G) \leq k$. Then, for $H \in L$, $\text{conn}_{\text{sub}}(H) = \max\{\text{conn}(G) | G \subseteq H\} \leq k$. $\square$

Remark

Connectivity Theorem II can be used to show that, e.g., “chains” of complete graphs K_n ($n \geq 1$) of the form



cannot be generated by a hyperedge-replacement grammar.

A *clique* of a graph H is any subgraph which is isomorphic to a complete graph K_i with i nodes, where $1 \leq i \leq |V_H|$ (see, e.g., Gibbons [Gi 85]).

4.5 Corollary (bounded clique size)

For each hyperedge-replacement language L , the members of which are simple graphs, there is a natural number k such that, for all $H \in L$, the size of the cliques in H , $\text{clique-size}(H)$, is bounded by k .

Proof

For a clique K of H , the connectivity is given by $\text{conn}(K) = |V_K| - 1$. Therefore,

$$\begin{aligned} \text{clique-size}(H) &= \max\{|V_K| \mid K \text{ is a clique of } H\} \\ &\leq \max\{\text{conn}(G) \mid G \subseteq H\} + 1 \\ &= \text{conn}_{\text{sub}}(H) + 1. \end{aligned}$$

By Connectivity Theorem II, there is a bound k such that $\text{conn}_{\text{sub}}(H) \leq k$ for all $H \in L$. Hence, $\text{clique-size}(H) \leq k + 1$ for all $H \in L$. $\square$

4.6 Definition (degree)

Let $H \in \mathcal{H}_C$ and $v \in V_H$. Then the *degree* of v in H , $\text{degree}_H(v)$, is the number of “tentacles” incoming in v of outgoing from v , i.e., the degree of v in H is defined to be the degree of v in the underlying graph. The *degree* of H , $\text{degree}(H)$, is the maximum degree of its nodes. Accordingly, the *minimum degree* of H , $\text{mindegree}(H)$, is the minimum of the degrees of its nodes.

4.7 Theorem (bounded minimum degree)

Let L be a hyperedge-replacement language of simple, non-empty hypergraphs. Then there is a natural number k such that, for all $H \in L$, there is a node $v \in V_H$ with $\text{degree}_H(v) \leq k$, i.e., $\text{mindegree}(H) \leq k$.

Proof

Let L be a hyperedge-replacement language. Then there exists a constant r such that L is of order r . Let p and q be the constants occurring in the Pumping Lemma. Then each $H \in L$ with $\text{size}(H) > p$ can be written as $H = \text{FIRST} \otimes \text{LINK} \otimes \text{LAST}$ such that $\text{size}(\text{LINK} \otimes \text{LAST}) \leq q$, $\text{size}(\text{frame}(\text{LINK})) \leq r$, LINK is non-trivial, and, for $k \in \mathbb{N}$, $\text{FIRST} \otimes \text{LINK}^k \otimes \text{LAST}$ is in L .

Since all graphs in L are simple and LINK is non-trivial, LINK must have at least one internal node. [If LINK possesses no internal nodes, then it must have edges between external nodes. Let $n = \frac{r(r-1)}{2} + 1$. Then LINK^n has at most r nodes and at least n terminal edges. Consequently, $\text{FIRST} \otimes \text{LINK}^n \otimes \text{LAST}$ possesses multiple edges or loops, a contradiction to the simplicity assumption.] Since $\text{size}(\text{LINK} \otimes \text{LAST}) \leq q$ and $\text{LINK} \otimes \text{LAST}$ is simple, for each internal node v in LINK , $\text{degree}(v) \leq q - 1$. Therefore, the minimum degree of H is bounded by $q - 1$.

For all $H \in L$ with $\text{size}(H) \leq p$, the minimum degree of H is bounded by $p - 1$. Therefore, $k = \max(p, q) - 1$ is a bound for the minimum degree of the hypergraphs in L . $\square$

Remark

Another proof based on derivation trees may look as follows. Let HRG be a proper hyperedge-replacement grammar generating L . Choose

$$k = \max\{|V_R| \mid R \text{ is a right-hand side of a production in } P \text{ or } R = Z\} - 1$$

and consider $H \in L(\text{HRG})$ with derivation $Z \xRightarrow{*} H$ and derivation tree T . If in the derivation no node-generating production is applied, then, by simplicity of H , $\text{mindegree}(H) \leq |V_Z| - 1 \leq k$. If at least one node-generating production is applied, we may find a subtree T_2 of T containing exactly one hyperedge the label of which is a right-hand side of a node-generating production and the source of which is the root of T_2 . T_2 is the derivation tree of a derivation $A^* \xRightarrow{*} R \xRightarrow{*} H_2$ for some $H_2 \in \mathcal{H}_T$. By choice of T_2 , for each internal node v of H_2 $\text{degree}_{H_2}(v) \leq \max\{|V_R| \mid (A, R) \in P\} - 1 \leq k$. Let T_1 be obtained from T by deleting T_2 up to the root of T_2 . Then T_1 is the derivation tree of a derivation $Z \xRightarrow{*} H_1$ for some hypergraph $H_1 \in \mathcal{H}_{N \cup T}$, which contains exactly one nonterminal hyperedge, say e , and this hyperedge is labeled by A . Replacing e by H_2 , we obtain H (up to isomorphism), again. Then H has a node $v \in V_H$ with $\text{degree}_H(v) \leq k$ because H_2 possesses a node with this property. Therefore, for $H \in L$, $\text{mindegree}(H) \leq k$.

Finally, we investigate hyperedge-replacement languages the members of which are simple, connected graphs. We say that a graph language L is of *bounded degree* if there exists an integer *bound* such that each node in each graph of L is of degree not exceeding *bound*.

Given a connected graph G , the *diameter* of G , denoted by $\text{diam}(G)$, is the maximal distance between two of its nodes. More precisely,

$$\text{diam}(G) = \max_{v, v' \in V_G} \min\{n \in \mathbb{N} \mid \text{there exists a path of length } n \text{ connecting } v \text{ and } v' \text{ in } G\}.$$

4.8 Theorem (diameter)

Let L be an infinite hyperedge-replacement language of bounded degree such that every $H \in L$ is a simple and connected graph. Then there exists a constant k depending only on L such that, for each natural number m there exists a graph $M \in L$ with

$$|V_M| \geq m \quad \text{and} \quad \text{diam}(M) \geq \frac{|V_M|}{k}.$$

Proof

Let L be a hyperedge-replacement language. Then there exists a constant r such that L is of order r . Let p and q be the constants occurring in the Pumping Lemma. Since L is infinite and all graphs in L are simple, there exists a graph $H \in L$ with $p < \text{size}(H) \leq p+q$ (see Corollary 2.5 in Chapter IV). By the Pumping Lemma, H can be written as $H = \text{FIRST} \otimes \text{LINK} \otimes \text{LAST}$ such that $\text{size}(\text{LINK} \otimes \text{LAST}) \leq q$, $\text{size}(\text{frame}(\text{LINK})) \leq r$, LINK is non-trivial, and, for $n \in \mathbb{N}$, $\text{FIRST} \otimes \text{LINK}^n \otimes \text{LAST}$ is in L .

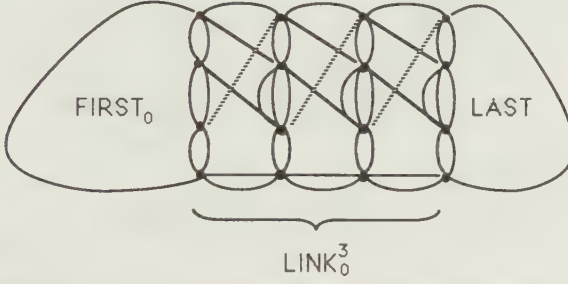
(a) If LINK possesses no internal nodes, then it must have edges between external nodes. Let $n = \frac{r(r-1)}{2} + 1$. Then LINK^n has at most r nodes and at least n terminal edges. Consequently, $\text{FIRST} \otimes \text{LINK}^n \otimes \text{LAST}$ possesses multiple edges or loops, a contradiction to the simplicity of the graphs in L .

(b) If LINK possesses internal nodes which are not connected with any external one, then $\text{FIRST} \otimes \text{LINK}^2 \otimes \text{LAST}$ possesses an isolated component, a contradiction to the connectedness of the graphs in L . [If the handle in LINK grips to external nodes, only, then $\text{LINK} \otimes \text{LAST}$ as well as $\text{FIRST} \otimes \text{LINK} \otimes \text{LAST}$ have an isolated component. If the handle in LINK grips to an internal node in LINK , then $\text{LINK}^2 \otimes \text{LAST}$ and $\text{FIRST} \otimes \text{LINK}^2 \otimes \text{LAST}$ possess an isolated component because the internal nodes of $\text{LINK} \otimes \text{LAST}$ are not connected with the internal nodes of the first version of LINK and the internal nodes of LINK are not connected with the external ones of LINK (serving as contact nodes to the node of FIRST).]

(c) If LINK possesses internal nodes and all are connected with any external one, but the handle of LINK grips to external nodes only, then there exist an n and an external node of LINK such that the degree of v in LINK^n as well as in $\text{FIRST} \otimes \text{LINK}^n \otimes \text{LAST}$ is greater than the given bound, a contradiction to the bounded-degree property.

(d) The last case to consider is that LINK possesses internal nodes, all of which are connected with some external ones, and the handle of LINK grips to some internal nodes. In this case, there exists an $n \geq 1$ such that, in LINK^n , the first tentacle grips to a node v_1 which is connected with the first external node ext_1 , the second tentacle grips to a node v_2 which is connected with the second external node ext_2 , ..., and the last tentacle grips to a node v_l which is connected with the last external node ext_l .

Let int denote the number of internal nodes in $LINK^n$ and d be the maximum of the distances between external nodes ext_i and the corresponding internal nodes v_i (v_i denotes the node which is endpoint of the i th tentacle of the handle in $LINK^n$) ($i = 1, \dots, l$).



Given a natural number m , consider the graph $M = FIRST \otimes LINK^{\bar{m} \cdot n} \otimes LAST$ where $\bar{m} = \max(m, p)$. Obviously,

$$|V_M| \geq \bar{m} \cdot int \geq m \quad \text{and} \quad diam(M) \geq \bar{m} \cdot d \geq \bar{m} \cdot n.$$

By choice of H , $|V_H| \leq size(H) \leq p + q$. Moreover, $H = FIRST \otimes LINK \otimes LAST$ and $|V_{LINK}| \leq size(LINK \otimes LAST) \leq q$. Therefore,

$$|V_M| \leq p + (\bar{m} \cdot n) \cdot q \leq 2q(\bar{m} \cdot n) \quad \text{and} \quad diam(M) \geq \bar{m} \cdot n \geq \frac{|V_M|}{2q}$$

□

Remarks

1. An analogous result for node-replacement languages in the sense of Janssens and Rozenberg can be found in [JR 81].
2. In Theorem 4.6, the requirement that L is of bounded degree is essential. E.g., the set $WHEEL$ of all wheels with at least 3 spokes is an infinite set of simple, connected graphs with unbounded degree, but bounded diameter.

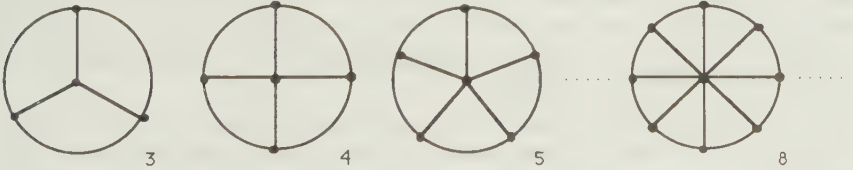


Figure 4.1: Wheels.

For a wheel $W(n)$ with n spokes, $degree(W(n)) = n$, $|V_{W(n)}| = n+1$, and $diam(W(n)) = 2$.

Therefore, there is no bound k such that, for all n , $\frac{|V_{W(n)}|}{diam(W(n))} \leq k$.

On the other hand, the set of wheels can be generated by the hyperedge-replacement grammar HRG_{WHEEL} with the axiom and the productions as given in Figure 4.2.

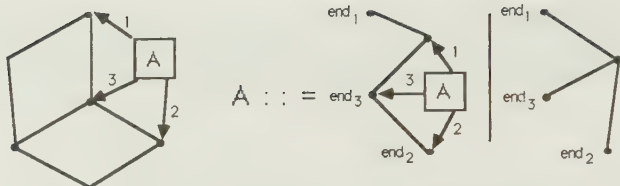


Figure 4.2: Axiom and productions of HRG_{WHEEL} .

4.9 Example (snails)

The set of all graphs of the form

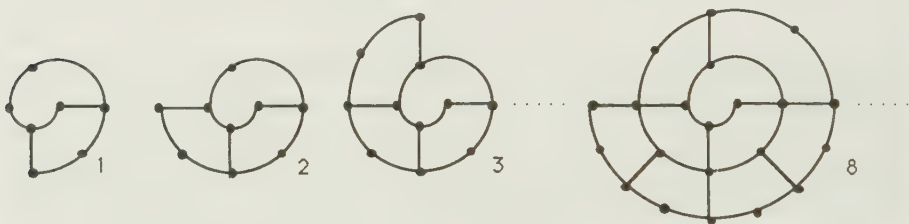


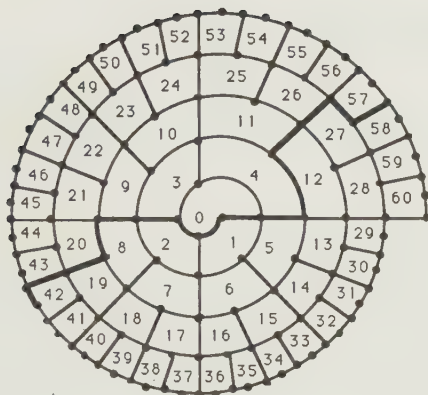
Figure 4.3: Snails.

called *snails*, cannot be generated by a hyperedge-replacement grammar although the connectivity and the degree are bounded (for each snail S , $conn(S) = 2$ and $degree(S) \leq 4$) and their growing with respect to the number of nodes and hyperedges is sublinear. This can be shown by the Pumping Lemma resp. Theorem 4.8.

The set of all snails forms an infinite graph language of bounded degree and every snail S is simple and connected. If the the set of snails would be a hyperedge-replacement language, then, by Theorem 4.8, there would exist a constant k such that, for each natural number m there exists a snail S with $|V_S| \geq m$ and $diam(S) \geq \frac{|V_S|}{k}$. We will show that, for snails, there does not exist such a constant.

For $n \geq 1$, let $S(n)$ denote the snail with n cells. Then $S(n)$ has $2n + 3$ nodes. Moreover, the diameter of $S(n)$ can be approximated as follows. Let i denote the i th node of the snail. Then the distance of the nodes i and j , $dist(i, j)$ is less or equal $dist(i, 1) + dist(j, 1)$.

A snail with i nodes possesses $sphere(i) = \lfloor \log(i - 3) \rfloor$ "spheres". To come from one sphere to the next lower one, one uses at most two steps to reach a lower sphere. (Each node is an outer node of a cell and the maximal distance from an outer node to an inner node of the cell is 2.) The maximal distance on the 0th sphere is 2. Therefore, the distance of a node v and the node 1, $dist(v, 1)$, is less or equal $2 \lfloor \log(v - 3) \rfloor + 2$ and the distance of two nodes v and v' is less or equal $2 \cdot dist(max(v, v'), 1)$.



Therefore,

$$\begin{aligned}
 \text{diam}(S(n)) &= \max_{v, v' \in V_{S(n)}} \text{dist}(v, v') \\
 &\leq 2 \max_{v, v' \in V_{S(n)}} \text{dist}(\max(v, v'), 1) = 2(\text{dist}(2n+5, 1)) \\
 &\leq 2(2\lfloor \log(2n+2) \rfloor + 2)
 \end{aligned}$$

Consequently, $\frac{|V_{S(n)}|}{\text{diam}(S(n))}$ grows beyond every bound. □

5. Generative Power of HRG's

This section is devoted to a subtler view of the order of a hyperedge-replacement grammar and the resulting generative power. In particular, hyperedge-replacement grammars the generated hypergraphs of which are ordinary graphs resp. string graphs, are objects of interest.

Generation of Graph Languages

In this subsection, we consider hyperedge-replacement grammars whose generated hypergraphs are ordinary graphs. In particular, edge-replacement grammars as considered in [Kr 79] and [HK 83+85] can be seen as such grammars. It can be shown that hyperedge-replacement grammars of order 2 and edge-replacement grammars generate the same graph languages. By the Pumping Lemma (resp. derived theorems), we are able to show that the classes of all hyperedge-replacement languages of order k , $k \in \mathbb{N}$, which generate graphs, form a proper hierarchy.

5.1 Definition (edge-replacement grammars and graph languages)

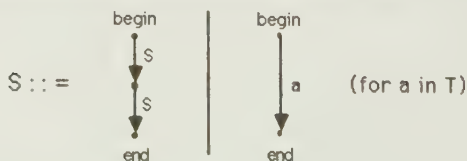
1. A hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is an *edge-replacement grammar*, if the right-hand sides of the productions as well as the axiom are $(1,1)$ -graphs. The class of all edge-replacement grammars is denoted by $\mathcal{ERG}$.
2. A hypergraph language L is said to be a *graph language* if all $H \in L$ are (multi-pointed) graphs. The class of all graph languages is denoted by $\mathcal{L}_{GRAPH}$. In particular, the class of all graph languages the graphs of which are $(1,1)$ -graphs is denoted by $\mathcal{L}_{GRAPH(1,1)}$.
3. The class of languages generated by an edge-replacement grammar is denoted by $\mathcal{ERL}$.

Remark

$$\mathcal{ERL} \subseteq \mathcal{HRL}_2 \cap \mathcal{L}_{GRAPH(1,1)}.$$

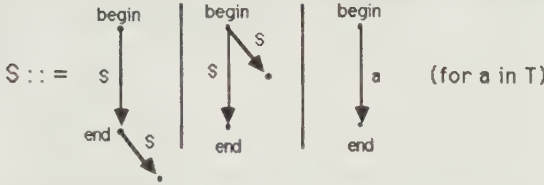
5.2 Examples (string graphs, trees and forests, collapsed trees, 2-trees, and maximal outerplanar graphs)

1. The edge-replacement grammar $ERG_{STRING} = (\{S\}, T, P, S^*)$ with the productions

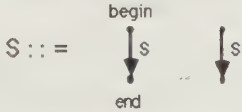


generates the set of all *string graphs* over T .

2. The edge-replacement grammar $ERG_{\text{TREE}} = (\{S\}, T, P, S^*)$ with the productions

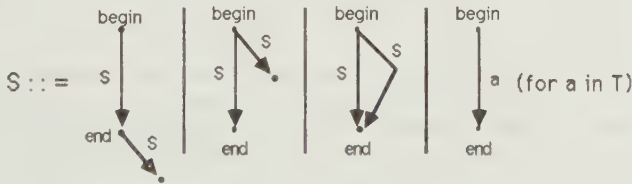


generates the set of all *trees* (with at least two nodes) over T . Adding the production



to ERG_{TREE} , we obtain a grammar ERG_{FOREST} , which generates the set of all *forests*.

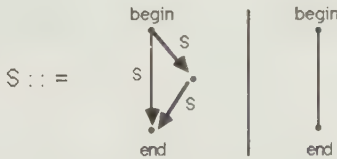
3. The set of all *collapsed trees* (with at least two nodes) can be generated by the grammar $ERG_{\text{CTREE}} = (\{S\}, T, P, S^*)$ with the productions



4. *2-trees* (see Rose [Ro 74]) are recursively defined as follows:

- (1) Each complete graph with two nodes is a 2-tree.
- (2) If e is an edge in a 2-tree, then the graph obtained by adding a new node v and edges from $s(e)$ to v and from v to $i(e)$ is a 2-tree.
- (3) Only the graphs obtained by (1) and (2) are 2-trees.

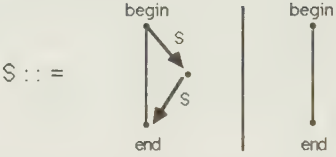
It is easy to see that the edge-replacement grammar $ERG_{2\text{TREE}}$ with the axiom S^* and the productions



simulates the recursive build-up of a 2-tree. Hence, $L(ERG_{2\text{TREE}})$ consists of all 2-trees.

5. Let ERG_{MOPS} be the edge-replacement grammar with the axiom S^* and the following

productions:



Following a simple characterization of maximal outerplanar graphs (see, e.g., Syslo [Sy 79]), it can be seen that $L(ERG_{MOPS})$ consists of all maximal outerplanar graphs. [Note that a *maximal outerplanar graph* is a triangulation of a polygon, such that by adding one edge between two nodes the obtained graph is no longer planar.]

As demonstrated in Chapter I (3.5), one may use hyperedge-replacement grammars to generate sets of graphs (e.g., sets of bipartite graphs). Given an arbitrary hyperedge-replacement grammar, it can be easily decided whether it generates only ordinary graphs.

5.3 Theorem (Graph Problem)

For hyperedge-replacement grammars, the Graph Problem is decidable, i.e., there is an algorithm for determining whether or not, given a hyperedge-replacement grammar HRG , $L(HRG) \in \mathcal{L}_{GRAPH}$ [resp. $L(HRG) \in \mathcal{L}_{GRAPH(1,1)}$].

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar. Without loss of generality, we may assume that HRG is reduced. Then $L(HRG) \in \mathcal{L}_{GRAPH}$ if and only if all terminal-labeled hyperedges occurring in a right-hand side of a production or in the axiom of HRG are $(1,1)$ -edges. Moreover, we have $L(HRG) \in \mathcal{L}_{GRAPH(1,1)}$ if and only if $L(HRG) \in \mathcal{L}_{GRAPH}$ and Z is a $(1,1)$ -graph. $\square$

In the following, we compare the power of graph-generating hyperedge-replacement grammars with the one of edge-replacement grammars. It turns out that hyperedge-replacement grammars of order 2 generating $(1,1)$ -graphs and edge-replacement grammars are of the same generative power. The basic idea for proving this statement is demonstrated in Example 5.4.

5.4 Example

The hyperedge-replacement grammar HRG with the axiom and the productions as in Figure 5.1 is a grammar of order 1. It generates the set of all trees with at least two nodes.

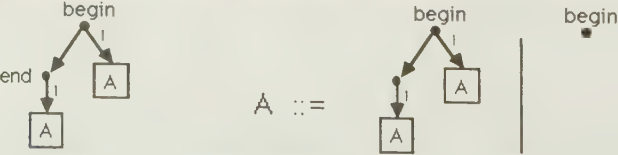


Figure 5.1: Axiom and productions of a grammar of order 1

The grammar can be modified to an edge-replacement grammar ERG by equipping each right-hand side with a second external node and each hyperedge with a second tentacle gripping to the additional node.

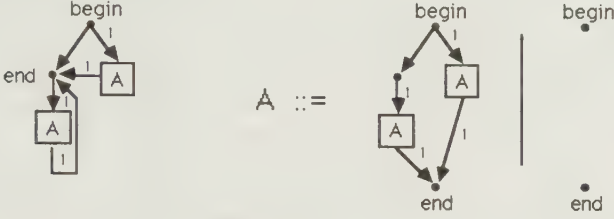


Figure 5.2: Axiom and productions of the modified grammar

5.5 Theorem

$$\mathcal{HRL}_2 \cap \mathcal{L}_{GRAPH(1,1)} = \mathcal{ERL}.$$

Proof

Let $HRG = (N, T, P, Z)$ be an arbitrary hyperedge-replacement grammar of order 2 and $L(HRG) \in \mathcal{L}_{GRAPH(1,1)}$. We may assume that HRG is typed and reduced. Moreover, we may assume that $P \neq \emptyset$. Otherwise, $HRG' = (N, T, P, Z')$ where Z' is any $(1,1)$ -graph over N is an edge-replacement grammar equivalent to HRG . Since HRG is of order 2, there is a type function $ltype$ such that, for all $A \in N$, $ltype(A) = (m, n)$ with $m + n \leq 2$. Using the Type-Modification Theorem I.4.4, the grammar can be modified in such a way that nonterminals of type $(2,0)$ or $(0,2)$ become nonterminals of type $(1,1)$. Therefore, we may assume that the nonterminal hyperedges are of type $(1,1)$, $(1,0)$, $(0,1)$, or $(0,0)$. Moreover, all terminal hyperedges are of type $(1,1)$. In the following, we construct a grammar in which all nonterminal hyperedges are of type $(1,1)$.

Let $H \in \mathcal{H}_C$ be a (m,n) -hypergraph, $m, n \leq 1$, and, for $e \in E_H$, $type(e) \in \{(0,0), (0,1), (1,0), (1,1)\}$.

Let H' be obtained from H according to the following rule: If $begin_H = \lambda$, add a node $begin$ and set $begin_{H'} = begin$, otherwise let $begin_{H'} = begin_H$; accordingly, if $end_H = \lambda$, add a node end and set $end_{H'} = end$, otherwise let $end_{H'} = end_H$. In this way, we obtain a $(1,1)$ -hypergraph H' .

Let $H_{mod} = (V'_H, E_H, s'_H, t'_H, l_H, begin'_{H'}, end'_{H'})$ be the $(1,1)$ -graph with

$$s'_H(e) = \begin{cases} s_H(e) & \text{for } type(e) = (1, n), 0 \leq n \leq 1 \\ begin_{H'} & \text{otherwise} \end{cases}$$

$$t'_H(e) = \begin{cases} t_H(e) & \text{for } type(e) = (m, 1), 0 \leq m \leq 1 \\ end_{H'} & \text{otherwise} \end{cases}.$$

Then H_{mod} is a $(1,1)$ -graph.

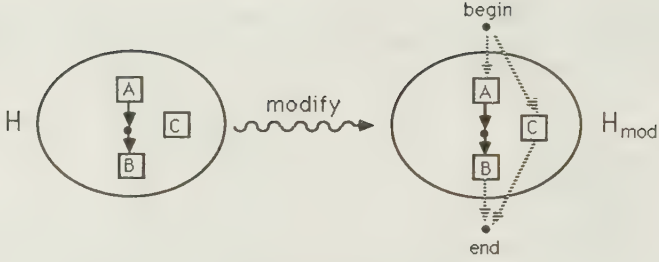


Figure 5.3: Modification of a $(0,0)$ -hypergraph.

Let $HRG_{\text{mod}} = (N, T, P_{\text{mod}}, Z_{\text{mod}})$ with $P_{\text{mod}} = \{(A, R_{\text{mod}}) \mid (A, R) \in P\}$. By assumption, $L(HRG) \in \mathcal{L}_{\text{GRAPH}(1,1)}$ and $L(HRG) \neq \emptyset$. Therefore, Z is a $(1,1)$ -hypergraph and Z_{mod} is a $(1,1)$ -graph. Moreover, the right-hand sides of productions in HRG_{mod} are $(1,1)$ -graphs. Hence, HRG_{mod} is an edge-replacement grammar. It remains to show that $L(HRG_{\text{mod}}) = L(HRG)$.

Claim 1: For each $A \in N$ and each derivation $A^* \xRightarrow{*} H$ in HRG with $H \in \mathcal{H}_T$, there is a derivation $(A^*)_{\text{mod}} \xRightarrow{*} H_{\text{mod}}$ in HRG_{mod} .¹

Proof of Claim 1 (by induction on the number of steps in the derivation): The statement is trivially true for one-step derivations. Suppose now that it is true for k -step derivations. Let $A^* \xRightarrow{k+1} H$ be a $(k+1)$ -step derivation in HRG and $H \in \mathcal{H}_T$. Then the derivation is of the form $A^* \xRightarrow{*} R' \xRightarrow{k} H$. If the first derivation step is a dummy, then $A^* \cong R'$, $R' \xRightarrow{k} H$ can be modified into $A^* \xRightarrow{k} H$, and, by inductive hypothesis, there is a derivation $(A^*)_{\text{mod}} \xRightarrow{*} H_{\text{mod}}$ in HRG_{mod} . Otherwise, a production $(A, R) \in P$ is applied to A^* yielding to R' isomorphic to R . Without loss of generality, $R' = R$. Consider now the fibres $l_R(e)^* \xRightarrow{*} H(e)$ of $R \xRightarrow{*} H$. By inductive hypothesis, for $e \in E_R$, there is a derivation $(l_R(e)^*)_{\text{mod}} \xRightarrow{*} H(e)_{\text{mod}}$ in HRG_{mod} . Joint embedding of these derivations into R_{mod} yields a derivation $R_{\text{mod}} \xRightarrow{*} H'$ in HRG_{mod} . It turns out that $H' = H_{\text{mod}}$ (up to isomorphism). Moreover, there is a direct derivation $(A^*)_{\text{mod}} \xRightarrow{*} R_{\text{mod}}$ through $(A, R_{\text{mod}}) \in P_{\text{mod}}$. Thus, there is a derivation $(A^*)_{\text{mod}} \xRightarrow{*} H_{\text{mod}}$ in HRG_{mod} .

Claim 2: For each $A \in N$ and each derivation $(A^*)_{\text{mod}} \xRightarrow{*} H'$ in HRG_{mod} with $H' \in \mathcal{H}_T$, there is a derivation $A^* \xRightarrow{*} H$ in HRG with $H_{\text{mod}} = H'$.

Proof of Claim 2 (by induction on the number of steps in the derivation): The statement is trivially true for one-step derivations. Suppose that the statement is true for k -step derivations. Let $(A^*)_{\text{mod}} \xRightarrow{k+1} H'$ be a $(k+1)$ -step derivation in HRG_{mod} and $H' \in \mathcal{H}_T$. Then the derivation decomposes into $(A^*)_{\text{mod}} \xRightarrow{*} R' \xRightarrow{k} H'$ for some $R' \in \mathcal{H}_{N \cup T}$. If the first derivation step is a dummy, then $(A^*)_{\text{mod}} \cong R'$, $R' \xRightarrow{k} H$ can be modified into

¹ A^* denotes an $l_{\text{type}(A)}$ -handle induced by A . $(A^*)_{\text{mod}}$ corresponds to the $(1,1)$ -handle induced by A

$(A^*)_{\text{mod}} \xRightarrow{k} H'$, and, by inductive hypothesis, there is a derivation $A^* \xRightarrow{*} H$ in HRG with $H_{\text{mod}} = H'$. Otherwise, a production $(A, R_{\text{mod}}) \in P_{\text{mod}}$ is applied. Then $R_{\text{mod}} \cong R'$ and $R' \xRightarrow{k} H'$ can be modified into a derivation $R_{\text{mod}} \xRightarrow{k} H'$. Consider now the fibres of $R_{\text{mod}} \xRightarrow{k} H'$. By inductive hypothesis, for each fibre $(l_R(e)^*)_{\text{mod}} \xRightarrow{k} H'(e)$ in HRG_{mod} , there is a corresponding derivation $l_R(e)^* \xRightarrow{*} H(e)$ in HRG where $H(e)_{\text{mod}} = H'(e)$. Joint embedding of the fibres into R yields a derivation $R \xRightarrow{*} H$ in HRG . It turns out that $H_{\text{mod}} = H'$ (up to isomorphism). Moreover, there is a direct derivation $A^* \Rightarrow R$ through $(A, R) \in P$. Composing the derivations, we get a derivation $A^* \xRightarrow{*} H$ in HRG . This completes the inductive step.

In particular, Claim 1 and 2 hold for $l(Z) \in N$. Consequently, for each derivation $Z \xRightarrow{*} H$ in HRG with $H \in \mathcal{H}_T$, there is a derivation $Z_{\text{mod}} \xRightarrow{*} H_{\text{mod}}$ in HRG_{mod} and, for each derivation $Z_{\text{mod}} \xRightarrow{*} H'$ in HRG_{mod} with $H' \in \mathcal{H}_T$, there is a derivation $Z \xRightarrow{*} H$ in HRG with $H_{\text{mod}} = H'$. Moreover, we have $H_{\text{mod}} = H$ for $H \in L(HRG)$ because $L(HRG) \in \mathcal{L}_{GRAPH}(1,1)$. Therefore, $L(HRG) = L(HRG_{\text{mod}})$. $\square$

For showing that arbitrary hyperedge-replacement grammars are more powerful than usual edge-replacement grammars and that the generative power increases with the number of tentacles a hyperedge may have, we give some illustrating examples.

5.6 Example (k -trees)

For $k \geq 1$, k -trees (see Rose [Ro 74]) are recursively defined as follows:

- (1) Each complete graph with k nodes is a k -tree.
- (2) Given a k -tree $T_k(n)$ on n nodes, a k -tree on $n+1$ nodes is obtained when the $(n+1)$ st node is made adjacent to each node of a k -clique in $T_k(n)$.

(Note that k -trees are generalizations of 2-trees, see Example 5.2.4.) For arbitrary but fixed positive k , one can easily find a hyperedge-replacement grammar $HRG_{k\text{TREE}}$ of order k generating the set of all k -trees. The grammar simulates the recursive build-up of a k -tree: As axiom we choose a complete graph with k nodes and an additional $(0, k)$ -hyperedge gripping to the k nodes (indicating that there is a k -clique). The “generating” production generates a new node *new*, k edges connecting *new* with the external nodes, and $k+1$ hyperedges indicating the old k -clique and the k arising k -cliques. The “terminating” production allows to remove a hyperedge.

5.7 Example (graphs of bandwidth $\leq k$)

The *bandwidth* of a graph H is the minimum integer k for which there exists a bijective mapping $f : V_H \rightarrow \{1, 2, \dots, |V_H|\}$ such that, for all $e \in E_H$,

$$|f(s_H(e)) - f(t_H(e))| \leq k.$$

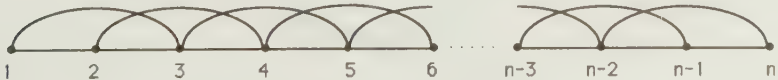


Figure 5.4: A graph of bandwidth ≤ 2 .

The hyperedge-replacement grammar HRG'_{2BW} with the axiom and the productions as shown in Figure 5.5 generates graphs of bandwidth ≤ 2 (with at least 2 nodes). Adding productions which replace dotted lines by usual ones or delete them yields a grammar HRG_{2BW} of order 2 which generates all graphs of bandwidth ≤ 2 having at least 2 nodes.

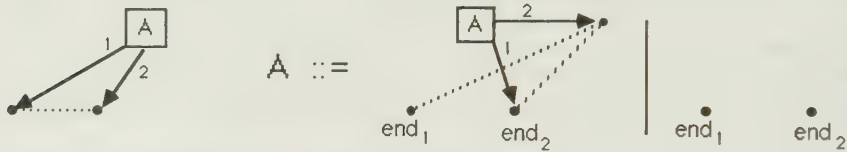


Figure 5.5: Axiom and productions of HRG'_{2BW} .

In a similar way, it can be shown that, for each k , there is a hyperedge-replacement grammar HRG_{kBW} of order k which generates all graphs of bandwidth $\leq k$ with at least k nodes. The axiom is a complete graph with k nodes and one hyperedge having k tentacles for gripping to the k nodes. The “generating” production produces a new node, connects it to the original nodes of the hyperedge, and produces a new hyperedge gripping to the “last” $k - 1$ nodes as well as to the new node; the “hyperedge-removing” production allows to remove the hyperedge; finally, the “edge-removing” production allows to remove an ordinary edge. In this way, all graphs of bandwidth $\leq k$ can be generated.

5.8 Example (graphs of cyclic bandwidth $\leq k$)

The *cyclic bandwidth* of a graph H is the minimum integer k for which there is a bijective mapping $f : V_H \rightarrow \{1, 2, \dots, |V_H|\}$, such that, for all $e \in E_H$,

$$\min \{|f(s_H(e)) - f(t_H(e))|, |V_H| - |f(s_H(e)) - f(t_H(e))|\} \leq k.$$

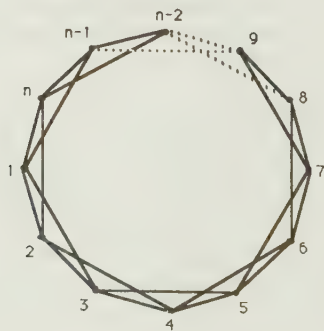


Figure 5.6: A graph of cyclic bandwidth ≤ 2 .

The hyperedge-replacement grammar HRG'_{2CBW} with the axiom and the productions as shown in Figure 5.7 generates graphs of cyclic bandwidth ≤ 2 (with at least 3 nodes). Adding productions which replace dotted lines by usual ones or delete dotted lines yields a grammar HRG_{2CBW} which generates all graphs of cyclic bandwidth ≤ 2 having at least 3 nodes.

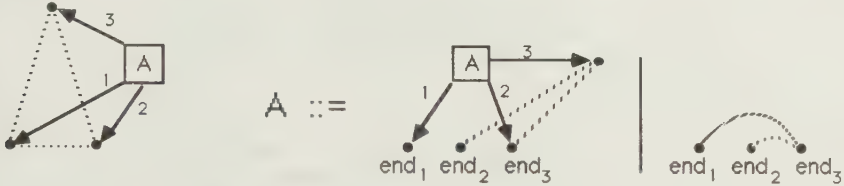


Figure 5.7: Axiom and productions of HRG'_{2CBW} .

Similarly, the set of all graphs of cyclic bandwidth $\leq k$ with at least $k + 1$ nodes can be generated by a hyperedge-replacement grammar HRG_{kCBW} of order $k + 1$: The axiom is a complete graph with $k + 1$ nodes and one hyperedge having $k + 1$ tentacles for gripping to the $k + 1$ nodes. The “node-generating” production produces a new node, connects it to the last k nodes (that are the nodes which are target of the $2nd, 3rd, \dots, (k + 1)st$ tentacle), and produces a new hyperedge gripping with its first tentacle to the first node, and with its other tentacles to the last k nodes. The “cycle-generating” production generates no new node but connects the first node (indicated by the first tentacle of the single hyperedge) with the last k nodes. Finally, an “edge-removing” production allows to remove an ordinary edge. In this way, all graphs of cyclic bandwidth $\leq k$ can be generated.

The “node-generating” production and the “cycle-generating” production look as follows.

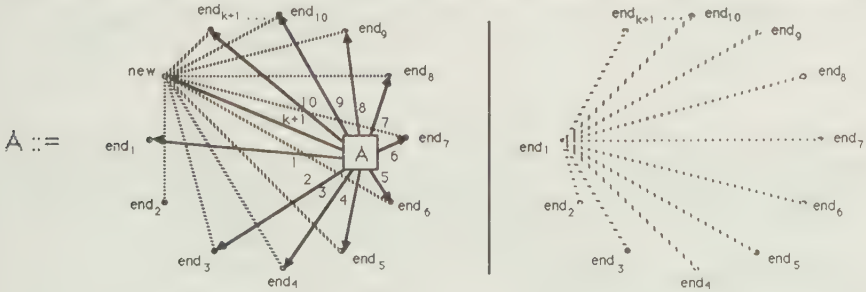
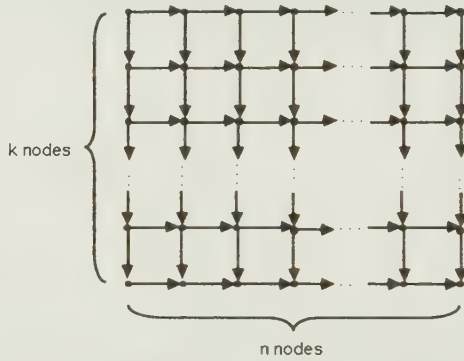


Figure 5.8: Productions of HRG_{kCBW} .

5.9 Example ($k \times n$ -grids)

For $k \geq 1$, the set of all $k \times n$ -grids (with $n \geq 2$) of the form



can be generated by the grammar $HRG_{kGRID} = (\{A\}, T, P, Z)$ with axiom and productions as given in Figure 5.9:

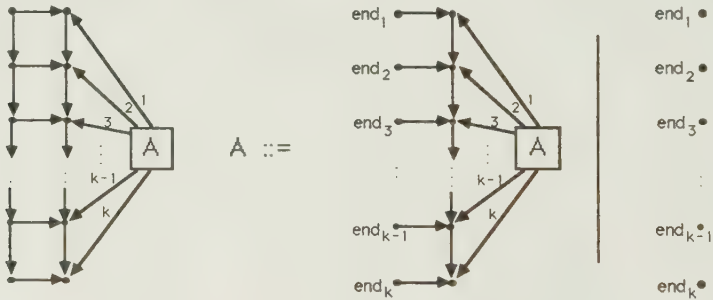


Figure 5.9: Axiom and productions of HRG_{kGRID}

As shown above, for each $k \geq 1$, the set of all k -trees can be generated by a hyperedge-replacement grammar of order k . On the other hand, the set of k -trees is of connectivity k . Therefore, it cannot be generated by a grammar of lower order.

5.10 Hierarchy Theorem

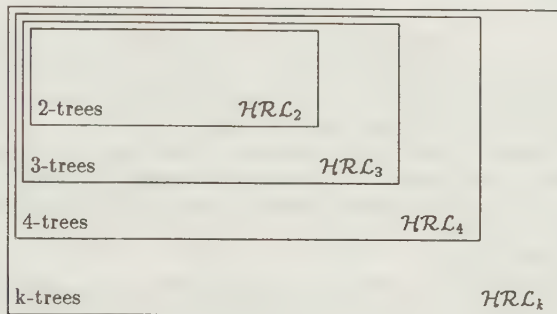
1. $HR\mathcal{L}_{k-1} \subset HR\mathcal{L}_k$ for all $k \geq 1$.
2. $HR\mathcal{L}_{k-1} \cap \mathcal{L}_{GRAPH} \subset HR\mathcal{L}_k \cap \mathcal{L}_{GRAPH}$ for all $k \geq 1$.

Proof

The set $kTREE$ of all k -trees consists of graphs, only. As shown in 5.6, $kTREE \in HR\mathcal{L}_k$. Because all k -trees are of connectivity k , the Connectivity Theorem (4.2) implies that $kTREE \notin HR\mathcal{L}_{k-1}$. □

Remark

As a consequence of the Hierarchy Theorem, the number of external nodes in the right-hand sides of productions cannot be bounded without reducing the generative power. The situation may be illustrated by the following diagram.



Finally, we give some applications of the Pumping Lemma resp. its derived theorems.

5.11 Theorem

For $k \geq 1$, the following graph languages are in $\text{HRL}_k - \text{HRL}_{k-1}$:

- (1) the set of all k -trees;
- (2) the set of all graphs of bandwidth $\leq k$;
- (3) the set of all graphs of cyclic bandwidth $\leq k - 1$;
- (4) the set BIP_k of all bipartite graphs in which one node set has k nodes (see I.3.5);
- (5) the set of all $k \times n$ -grids.

Proof

As shown in the Examples 5.6–5.9 of this section and 3.5 of Chapter I, for $k \geq 1$, the interesting sets of graphs can be generated by a hyperedge-replacement grammar of order k . On the other hand, k -trees are of connectivity k , there is an infinite number of graphs of bandwidth k whose connectivity is k , and an infinite number of graphs of cyclic bandwidth $k - 1$ whose connectivity is $\geq k$; moreover, all complete bipartite graphs of BIP_k are of connectivity k . By the Connectivity Theorem, the languages are not in HRL_{k-1} . Finally, making directly use of the Pumping Lemma, it can be shown that the set of all $k \times n$ -grids is not in HRL_{k-1} . $\square$

5.12 Theorem

The following sets of graphs cannot be generated by a hyperedge-replacement grammar:

- (1) all graphs;
- (2) all complete graphs;
- (3) all bipartite graphs;

- (4) all Eulerian graphs;
- (5) all Hamiltonian graphs;
- (6) all regular graphs;
- (7) all planar graphs;
- (8) all square grids;
- (9) all snails.

Proof

(2) The set of all complete graphs cannot be generated because the connectivity as well as the minimum degree of the language are unbounded.

(1) The set of all graphs cannot be generated because each complete graph is a graph and the set of complete graphs is of unbounded connectivity and unbounded minimum degree.

(3) The set of bipartite graphs cannot be generated because each complete bipartite graph is bipartite and the set of complete bipartite graphs is of unbounded connectivity and unbounded minimum degree.

(4) The set of all Eulerian graphs cannot be generated because each complete graph K_{2n+1} with odd number of nodes is connected and every node in it has even degree. By a characterization, we can conclude that these graphs are Eulerian. Therefore, the connectivity of the language is unbounded.

(5) The set of all Hamiltonian graphs cannot be generated because each complete graph is Hamiltonian. Therefore, the connectivity of the language is unbounded.

(6) By definition, a graph G is regular if there is some k such that $\text{degree}(v) = k$ for all nodes of G . Therefore, the minimum degree of the set of regular graphs is unbounded.

(7) The set of all planar graphs cannot be generated because the tree-width of planar graphs is unbounded. For instance, an $n \times n$ -grid has tree-width n (see Robertson and Seymour [RS 86]). On the other hand, it is easy to show that, for each hyperedge-replacement grammar HRG , the tree-width is bounded, i.e., there is a natural number k such that, for all $H \in L(HRG)$, H is of tree-width $\leq k$. (In Lautemann [La 88a], it is shown that, for each hyperedge-replacement grammar HRG , there is a natural number k such that, for all $H \in L(HRG)$, H is k -decomposable. Since Lautemann's notion of " k -decomposability" corresponds to the notion of "tree-width" introduced by Robertson and Seymour, the boundedness of the tree-width follows directly from the boundedness result given in [La 88a].)

(8) The set of all square grids cannot be generated because the number of nodes (and edges) grows too fast (see 2.8). Moreover, Theorem 4.8 can be used to prove the statement. For an $n \times n$ -grid $G(n)$, $|V_{G(n)}| = n^2$ and $\text{diam}(G(n)) = 2(n-1)$. Therefore, the quotient of $|V_{G(n)}|$ and $\text{diam}(G(n))$ grows beyond every bound.

(9) The set of all snails cannot be generated by a hyperedge-replacement grammar although all snails are of bounded connectivity, bounded degree, and the growing of the number of nodes and edges is linear. The reason is that the quotient of $|V_{S(n)}|$ and $\text{diam}(S(n))$ grows beyond every bound (see 4.9). □

Generation of String-Graph Languages

As demonstrated in Chapter I (3.6), hyperedge-replacement grammars can be used to generate languages of “string-like structure”. This seems to be quite attractive, because it provides a context-free mechanism for generating certain non-context-free string languages.

5.13 Definition (string-graph grammars and languages)

1. A hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is said to be a *context-free string-graph grammar*, if the right-hand sides of the productions as well as the axiom Z are string graphs.
2. A set of string graphs (closed under isomorphism) is said to be a *string-graph language*. The class of all string-graph languages is denoted by $\mathcal{L}_{STRING}$. The class of all string-graph languages generated by a context-free string-graph grammar is denoted by $\mathcal{CFL}$.

Remark

Each context-free string grammar $G = (N, T, P, S)$ (without λ -rules) induces a context-free string-graph grammar $G^* = (N, T, P^*, S^*)$ with $P^* = \{(A, r^*) \mid (A, r) \in P\}$ such that the string-graph language $L(G^*)$ generated by G^* is obtained from the string language $L(G)$ generated by G just by string-to-string-graph transformation:

$$L(G^*) = \{w^* \mid w \in L(G)\}.$$

In this sense, hyperedge-replacement grammars generalize context-free string grammars.

5.14 Example

The string-graph language $WW = \{(ww)^* \mid w \in T^+\}$ can be generated by the hyperedge-replacement grammar $HRG_{WW} = (\{S, A\}, T, P, S^*)$ of order 4 with the productions

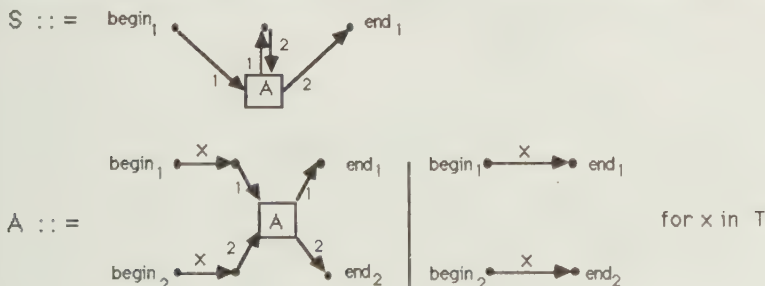


Figure 5.10: Productions of HRG_{WW} .

Note that the corresponding string language is not context-free.

5.15 Theorem (String-Graph Problem)

For hyperedge-replacement grammars of order r (for some $r \in \mathbb{N}$), the String-Graph Problem is decidable, i.e., there is an algorithm which determines, for a given a hyperedge replacement grammar $HRG \in \mathcal{HRG}_r$, whether $L(HRG) \in \mathcal{L}_{STRING}$.

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar of order r . By the Typification Theorem I.4.3, we may assume that HRG is typed. It can be shown that the property STR given by

$STR(H)$ if and only if H is a string graph

is HRG_r -compatible in the sense of Definition 3.1 of Chapter V. Therefore, by Theorem 3.7 in Chapter V, it is decidable whether or not STR holds for all $H \in L(HRG)$; i.e., it is decidable whether all $H \in L(HRG)$ are string graphs. □

A Normal-Form Theorem

We now prove a normal-form theorem for hyperedge-replacement grammars which generate string graphs, only. Informally, it states that all hyperedge-replacement grammars generating string graphs are equivalent to grammars with restrictions on the forms of productions.

5.16 Example

The string-graph language $ABCD = \{(a^m b^n c^m d^n)^* | m, n \geq 1\}$ can be generated by the hyperedge-replacement grammar $HRG_{ABCD} = (\{S, A, B\}, T, P, S^*)$ of order 4 with the productions

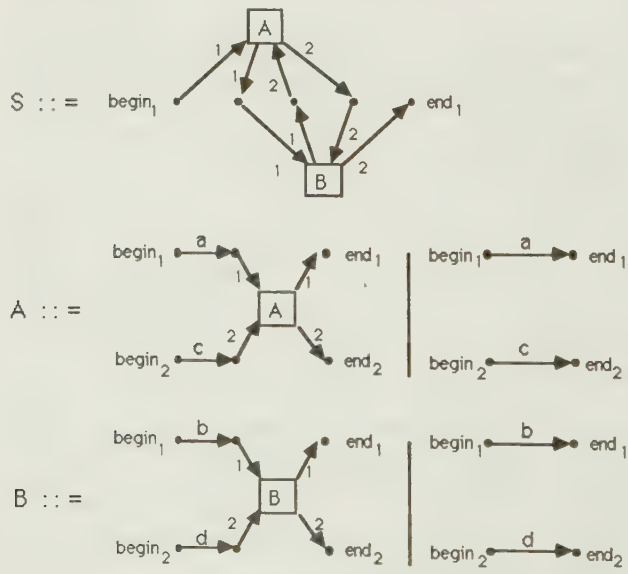


Figure 5.11: Productions of HRG_{ABCD} .

5.17 Definition (compressed bundles of string graphs)

1. A hypergraph $H \in \mathcal{H}_C$ is said to be *well-proportioned* if $|begin_H| = |end_H|$ and, for $e \in E_H$, $|s_H(e)| = |t_H(e)|$.
2. A well-proportioned hypergraph H is called a *bundle of string graphs* if there are string graphs $S_1 \dots S_k$ ($k \geq 1$) such that H is isomorphic to the (k, k) -hypergraph $\sum_{i=1}^k S_i$ given by $(\sum_{i=1}^k V_{S_i}, \sum_{i=1}^k E_{S_i}, s, t, l, begin, end)$ with $s(e) = s_{S_i}(e)$, $t(e) = t_{S_i}(e)$, $l(e) = l_{S_i}(e)$ for $i \in \{1, \dots, k\}$ and $e \in E_{S_i}$, $begin = begin_{S_1} \dots begin_{S_k}$, $end = end_{S_1} \dots end_{S_k}$.
3. A well-proportioned hypergraph H is said to be a *compressed bundle of string graphs* if $unfold(H)$ is a bundle of string graphs. The operation *unfold* works as follows: For a (k, k) -edge $e \in E_H$ with label A , $unfold(e)$ denotes the (k, k) -hypergraph X with $V_X = \{begin_i | i = 1, \dots, k\} \cup \{end_i | i = 1, \dots, k\}$, $E_X = \{e_i | i = 1 \dots k\}$, $s_X(e_i) = begin_i$, $t_X(e_i) = end_i$, $l_X(e_i) = A_i$, $begin_X = begin_1 \dots begin_k$, and $end_X = end_1 \dots end_k$. For a (k, k) -hypergraph H , $unfold(H)$ denotes the (k, k) -hypergraph $REPLACE(H, repl)$ with $repl(e) = unfold(e)$ for $e \in E_H$.

Remark

The multi-pointed hypergraphs given in Example 5.16 are compressed bundles of string graphs. In particular, the right-hand side of the first production is a compressed string graph and the right-hand sides of the third and fifth production are bundles of string graphs.

5.18 Theorem (Normal-Form Theorem)

For each hyperedge-replacement grammar HRG generating a string-graph language, there is an equivalent grammar HRG_{norm} in which the axiom is a string graph and all right-hand sides of productions are compressed bundles of string graphs.

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar with $L(HRG) \in \mathcal{L}_{STRING}$. We may assume that HRG is typed, wellformed, and reduced. If $P = \emptyset$, then choose any string graph as axiom. Otherwise, by reducedness of HRG , for each $A \in N$, there is a derivation $Z \xrightarrow{*} G$ where G is terminal up to one hyperedge e with label A . Moreover, there is a derivation $A^* \xrightarrow{*} F$ with $F \in \mathcal{H}_T$. Embedding of the derivation into G yields a derivation $G \xrightarrow{*} H$ with $H \in \mathcal{H}_T$. Since all terminal hypergraphs generated by HRG are string graphs, H is a string graph. The situation is depicted in Figure 5.12.

By well-formedness of HRG , the image of F in H is isomorphic to F . Therefore, we may assume that F is a subhypergraph of H . Since H is a string graph,

- $d_F^-(v) = d_F^+(v) = 1$ for $v \in INT_F$,
- $0 \leq d_F^-(v) \leq 1$ and $0 \leq d_F^+(v) \leq 1$ for $v \in EXT_F$.

(For a hypergraph F and a node $v \in V_F$, $d_F^-(v)$ and $d_F^+(v)$ denote the outdegree and the indegree of v , respectively.) Moreover, for two derivations $A^* \xrightarrow{*} F$, $A^* \xrightarrow{*} F'$ with $F, F' \in \mathcal{H}_T$, corresponding external nodes have the same degree. (Otherwise, the language would contain a hypergraph which is not a string graph.)

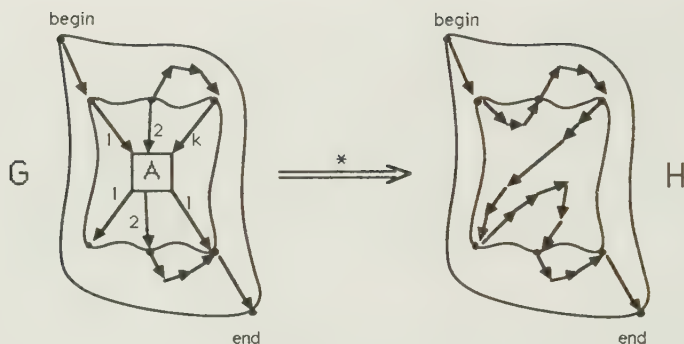


Figure 5.12: A derivation of a string graph

Two types of external nodes play a specific role:

- (1) external nodes for which $d_F^-(v) = d_F^+(v) = 0$,
- (2) external nodes for which $d_F^-(v) = d_F^+(v) = 1$.

In the first case, the derivation $A^* \xRightarrow{*} F$ does not produce any edge incident with v . The same holds for each other derivation $A^* \xRightarrow{*} F'$ to a terminal graph F' . In this sense, v is irrelevant in each derivation starting from A^* and one may be tempted to cut the tentacle of the hyperedge e of A^* gripping to v . In the second case, the derivation $A^* \xRightarrow{*} F$ produces one edge incident to v and one edge incident from v . The same holds for each other derivation $A^* \xRightarrow{*} F'$ to a terminal graph F' . Therefore, in all derivations starting from A^* , the required incoming edge as well as the required outgoing edge for v is generated. Considering a derivation $Z \xRightarrow{*} G$ with some hyperedge e with label A , the hyperedges $e' \in E_G - \{e\}$ do not produce any edge incident with v . In this sense, v is irrelevant for them and one may attempt to cut the tentacle of the hyperedge e of A^* gripping to v , to delete the node, and to generate it at the moment when the first edge for it shall be generated.

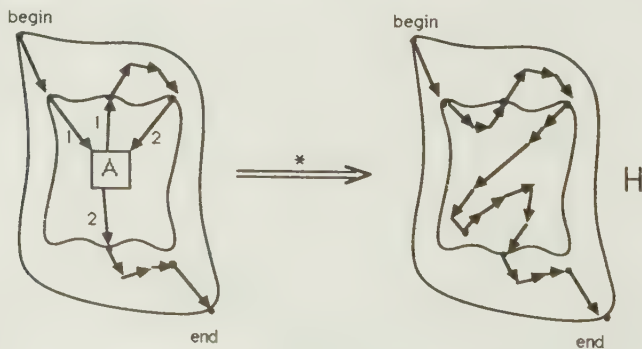


Figure 5.13: A derivation of a string graph

This observation is the basis of the “normalization” of HRG . We proceed as follows:

(1) For $A \in N$, choose a hypergraph $G \in \mathcal{H}_{N \cup T}$ which is terminal up to one hyperedge e with label A such that $Z \xrightarrow{*} G$ and a hypergraph $F \in \mathcal{H}_T$ such that $A^* \xrightarrow{*} F$. [By reducedness of HRG , we can construct effectively hypergraphs satisfying these properties. The special choices of G and F do not influence the following constructions.] Let $G \xrightarrow{*} H$ be the derivation obtained by embedding the derivation $A^* \xrightarrow{*} F$ into G . Then H is a string graph, i.e., there is path p from begin_H to end_H . On the other hand, $H = \text{REPLACE}(G, e \mapsto F)$. Running through the path, one finds a sequence of paths $p_1, q_1, p_2, \dots, p_k, q_k, p_{k+1}$ such that $p_1, p_2, \dots, p_{k+1}$ are non-empty paths in $G - \{e\}$ (except perhaps p_1 and p_{k+1}) and $q_1, \dots, q_k$ are non-empty paths in (the image of) F in H . For $i = 1, \dots, k$, let u_i be the initial endpoint and v_i the final endpoint of the path q_i . Obviously, the nodes $u_1, \dots, u_k, v_1, \dots, v_k$ are external nodes of F . Moreover, they are pairwise different.

Now define a singleton $\text{mod}(A)$ by $\text{mod}(A) = (\text{EXT}_F, \{e\}, s, t, l, \text{begin}_F, \text{end}_F)$, where $s(e) = u_1 \dots u_k$, $t(e) = v_1 \dots v_k$, and $l(e) = A$.

(2) For a production (A, R) in P , let

- $R' = \text{REPLACE}(R, \text{repl})$ where $\text{repl} : E_R \rightarrow \mathcal{H}_C$ is given by $\text{repl}(e) = \text{mod}(l_R(e))$,
- $R'' = (V_{R'}, E_{R'}, s_{R'}, t_{R'}, l_{R'}, u'_1 \dots u'_k, v'_1 \dots v'_k)$ where and $u'_1, \dots, u'_k, v'_1, \dots, v'_k$ are the external nodes in R' corresponding to $u_1, \dots, u_k, v_1, \dots, v_k$ in F ,
- $R_{\text{mod}} = R'' - \{v \in V_{R''} \mid d_{R''}^-(v) = d_{R''}^+(v) = 0\}$

(3) For $H \in \mathcal{H}_C$, let

- $H' = \text{REPLACE}(H, \text{repl})$ where $\text{repl} : E_H \rightarrow \mathcal{H}_C$ is given by $\text{repl}(e) = \text{mod}(l_H(e))$,
- $H_{\text{norm}} = H' - \{v \in V_{H'} \mid d_{H'}^-(v) = d_{H'}^+(v) = 0\}$.

Now let $HRG_{\text{norm}} = (N, T, P_{\text{norm}}, Z_{\text{norm}})$ be the hyperedge-replacement grammar with $P_{\text{norm}} = \{(A, R_{\text{mod}}) \mid (A, R) \in P\}$.

By construction, HRG_{norm} is in normal form. (Each right-hand side R_{mod} is well-proportioned. Moreover, if v is a begin-node in R_{mod} , then $d_{R_{\text{mod}}}^-(v) = 0$ and $d_{R_{\text{mod}}}^+(v) = 1$; if v is an end-node, then $d_{R_{\text{mod}}}^-(v) = 1$ and $d_{R_{\text{mod}}}^+(v) = 0$, and if v is an internal node, then $d_{R_{\text{mod}}}^-(v) = 1$ and $d_{R_{\text{mod}}}^+(v) = 1$. It turns out that R_{mod} is a compressed bundle of string graphs.)

Thus, it remains to show that $L(HRG_{\text{norm}}) = L(HRG)$.

To show that $L(HRG) \subseteq L(HRG_{\text{norm}})$, we use the following observation: Whenever a production (A, R) is used in a derivation $Z \xrightarrow{*} H$, the modified version of the production is used in the derivation $Z_{\text{norm}} \xrightarrow{*} H_{\text{norm}}$.

Claim 1: Let $Z \xrightarrow{*} H$ be a derivation in HRG . Then there is a derivation $Z_{\text{norm}} \xrightarrow{*} H_{\text{norm}}$ in HRG_{norm} where H_{norm} is the normalized version of H .

Proof of Claim 1 (by induction on the number of steps in the derivation): The statement is trivial for zero-step derivations: In this case, $Z \cong H$ and $Z_{\text{norm}} \cong H_{\text{norm}}$. Suppose that it is true for derivations of up to k steps. Let $Z \xrightarrow{*} H$ be a $(k+1)$ -step derivation in HRG . Then the derivation is of the form $Z \xrightarrow{k} G \xrightarrow{(A, R)} H$, where $(A, R) \in P$. Let e denote the hyperedge in G with label A replaced in the last derivation step. By inductive hypothesis, there is a derivation $Z_{\text{norm}} \xrightarrow{*} G_{\text{norm}}$ in HRG_{norm} , where G_{norm} is the normalized version

of G . In G , (A, R) is applied to e , i.e., $l_G(e) = A$ and $\text{type}(e) = \text{type}(R)$. In G_{norm} , $\text{type}(e) = \text{type}(R_{\text{mod}})$, i.e., the production $(A, R_{\text{mod}}) \in P_{\text{norm}}$ can be applied to e in G_{norm} leading to a multi-pointed hypergraph $\tilde{H}$. It turns out that $\tilde{H} = H_{\text{norm}}$ (up to isomorphism). Thus, there is a derivation $Z_{\text{norm}} \xrightarrow{*} H_{\text{norm}}$ in HRG_{norm} , where H_{norm} is the normalized version of H . This completes the inductive proof of Claim 1.

Claim 2: Let $Z_{\text{norm}} \xrightarrow{*} \tilde{H}$ be a derivation in HRG_{norm} . Then there is a derivation $Z \xrightarrow{*} H$ in HRG such that $H_{\text{norm}} = \tilde{H}$ (up to isomorphism).

Proof of Claim 2 (by induction on the number of steps): For zero-step derivations, the statement is trivial: In this case, $Z_{\text{norm}} \cong \tilde{H}$, $Z \cong H$, and $H_{\text{norm}} \cong Z_{\text{norm}} \cong \tilde{H}$. Suppose that the statement is true for k -step derivations and let $Z_{\text{norm}} \xrightarrow{*} \tilde{H}$ be a $(k+1)$ -step derivation in HRG_{norm} . Then the considered derivation is of the form $Z_{\text{norm}} \xrightarrow[k]{P_{\text{norm}}} \tilde{G} \xrightarrow{(A, \tilde{R})} \tilde{H}$

where $(A, \tilde{R}) \in P_{\text{norm}}$. By inductive hypothesis, there is a derivation $Z \xrightarrow{*} G$ in HRG where $G_{\text{norm}} \cong \tilde{G}$. Let e denote the hyperedge in $\tilde{G}$ with label A replaced in the last derivation step. Then $\text{type}(e) = \text{type}(R_{\text{mod}})$. In G , $\text{type}(e) = \text{type}(R)$. Thus, the production $(A, R) \in P$ can be applied to e in G leading to a multi-pointed hypergraph H . It turns out that $H_{\text{norm}} = \tilde{H}$ (up to isomorphism). Hence we can conclude that there is a derivation $Z \xrightarrow{*} H$ in HRG with the desired properties. This completes the inductive proof of Claim 2.

Note that, for string graphs $H \in \mathcal{H}_T$, $H_{\text{norm}} \cong H$. Since $L(HRG) \in \mathcal{L}_{\text{STRING}}$, Claim 1 and 2 imply $L(HRG) = L(HRG_{\text{norm}})$. $\square$

5.19 Corollary (even order)

Let HRG be a hyperedge-replacement grammar of order r and $L(HRG) \in \mathcal{L}_{\text{STRING}}$. Then there is an equivalent grammar HRG_{norm} the order of which is even and $\leq r$.

Proof

Let $HRG = (N, T, P, Z)$ be a hyperedge-replacement grammar with $L(HRG) \in \mathcal{L}_{\text{STRING}}$ and $HRG_{\text{norm}} = (N, T, P_{\text{norm}}, Z_{\text{norm}})$ the corresponding normal form. By Theorem 5.18, HRG_{norm} is equivalent to HRG . Moreover, if HRG is of order r , then $|EXT_R| \leq r$ for all $(A, R) \in P$. In HRG_{norm} , $|EXT_{R_{\text{mod}}}|$ is even and $|EXT_{R_{\text{mod}}}| \leq |EXT_R|$ for all $(A, R_{\text{mod}}) \in P_{\text{norm}}$. Therefore, HRG_{norm} is a grammar the order of which is even and $\leq r$. $\square$

As, e.g., the examples given in 5.14 and 5.16 show, hyperedge-replacement grammars can be used to generate languages of “string-like structure”. This seems to be quite attractive, because it provides a context-free mechanism for generating certain non-context-free string languages. Denoting the class of all string-graph languages by $\mathcal{L}_{\text{STRING}}$, we get an infinite hierarchy. The power of hyperedge-replacement grammars generating languages from $\mathcal{L}_{\text{STRING}}$ increases whenever the considered order increases by more than one. Somewhat amazing, the grammars of order $2r$ and $2r+1$ for $r \in \mathbb{N}$ generate the same subclass of $\mathcal{L}_{\text{STRING}}$ (see Theorem 5.22).

First we will show that the power of hyperedge-replacement grammars of order 2 generating string-graph languages equals the power of context-free string-graph grammars.

5.20 Theorem

$$\mathcal{HRL}_2 \cap \mathcal{L}_{\text{STRING}} = \mathcal{CFL}.$$

Proof

By Definition 5.13, $\mathcal{CFL} \subseteq \mathcal{HRL}_2 \cap \mathcal{L}_{\text{STRING}}$. Conversely, let $L \in \mathcal{HRL}_2 \cap \mathcal{L}_{\text{STRING}}$. Then there is a proper hyperedge-replacement grammar HRG of order 2 generating L . By the Normal-Form Theorem, there is an equivalent grammar HRG_{norm} in normal form, i.e., the axiom is a string graph and the right-hand sides of the productions are compressed bundles of string graphs. Since HRG_{norm} is of order 2, the compressed bundles of string graphs in the right-hand sides of productions are string graphs. Therefore, HRG_{norm} is a context-free string-graph grammar, i.e., $L = L(HRG) = L(HRG_{\text{norm}}) \in \mathcal{CFL}$. $\square$

5.21 Example

For $k \geq 1$ and $a_i, b_i \in T$ ($i = 1, \dots, k$), the string-graph language

$$L_{2k} = \{(a_1^n b_1^n a_2^n b_2^n \dots a_k^n b_k^n)^* \mid n \geq 1\}$$

can be generated by a hyperedge-replacement grammar HRG_{2k} of order $2k$.

For $k = 6$, the productions look as follows:

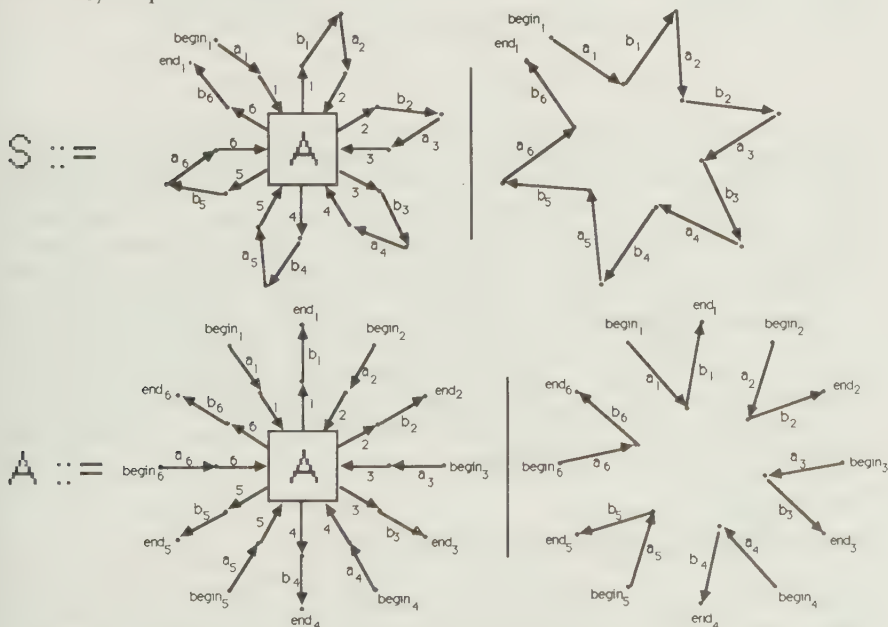


Figure 5.14: Productions of the grammar HRG_{12} .

In a similar way, it can be shown that, for $k \geq 1$ and $a_i, b_i \in T$ ($i = 1, \dots, k$), the string-graph language

$$L_{2k-1} = \{(a_1^n b_1^n a_2^n b_2^n \dots a_k^n)^* | n \geq 1\}$$

can be generated by a hyperedge-replacement grammar HRG_{2k} of order $2k$. For $k = 2$, $a_1 = a$, $b_1 = b$, and $a_2 = c$, the productions are given in I.3.6.

5.22 Theorem (Hierarchy Theorem)

$$\begin{aligned} \mathcal{HRL}_{2k} \cap \mathcal{L}_{STRING} &= \mathcal{HRL}_{2k+1} \cap \mathcal{L}_{STRING}, \\ \mathcal{HRL}_{2k+1} \cap \mathcal{L}_{STRING} &\subset \mathcal{HRL}_{2k+2} \cap \mathcal{L}_{STRING} \end{aligned} \quad \text{for all } k \in \mathbb{N}.$$

Proof

By definition of the order, we have $\mathcal{HRL}_r \subseteq \mathcal{HRL}_{r+1}$ for all $r \in \mathbb{N}$. Consequently, $\mathcal{HRL}_r \cap \mathcal{L}_{STRING} \subseteq \mathcal{HRL}_{r+1} \cap \mathcal{L}_{STRING}$. Therefore, it remains to show the equality in the case that r is even and the inequality in the case that r is odd.

Case 1: $r = 2k$. Let $L \in \mathcal{HRL}_{2k+1} \cap \mathcal{L}_{STRING}$. Then there is a hyperedge-replacement grammar HRG of order $2k+1$ generating L . By Corollary 5.19, there is an equivalent grammar HRG_{norm} the order of which is even and $\leq 2k+1$. Therefore, $L \in \mathcal{HRL}_{2k} \cap \mathcal{L}_{STRING}$.

Case 2: $r = 2k+1$. As Example 5.21 shows, the string-graph language

$$L_{2k+2} = \{(a_1^n b_1^n a_2^n b_2^n \dots a_{k+1}^n b_{k+1}^n)^* | n \geq 1\}$$

can be generated by a hyperedge-replacement grammar of order $2k+2$, i.e., $L_{2k+2} \in \mathcal{HRL}_{2k+2} \cap \mathcal{L}_{STRING}$. By the Pumping Lemma, it can be shown that L_{2k+2} cannot be generated by a grammar of order $2k+1$, i.e., $L_{2k+2} \notin \mathcal{HRL}_{2k+1} \cap \mathcal{L}_{STRING}$. This can be seen as follows: Suppose L_{2k+2} were a hyperedge-replacement language of order $2k+1$. Let p and q (with $p \geq q$) be the constants of the Pumping Lemma. Consider the string graph $H = (a_1^p b_1^p a_2^p b_2^p \dots a_{k+1}^p b_{k+1}^p)^*$. Let $H = \text{FIRST} \otimes \text{LINK} \otimes \text{LAST}$ satisfy the conditions of the Pumping Lemma. Then as $\text{size}(\text{frame}(\text{LINK})) \leq 2k+1$, $\text{LINK} \otimes \text{LAST}$ can contain at most a bundle of k string-graphs. As $\text{size}(\text{LINK} \otimes \text{LAST}) \leq q$, each connected component of $\text{LINK} \otimes \text{LAST}$ contains at most two different labels. Otherwise $\text{size}(\text{LINK} \otimes \text{LAST}) \geq p+2 > q$. Therefore, $\text{LINK} \otimes \text{LAST}$ contains at most $2k$ different labels. Since H contains $2k+2$ different labels, there are some labels of H which do not occur in LINK . Therefore, $H = \text{FIRST} \otimes \text{LINK}^0 \otimes \text{LAST} \notin L_{2k+2}$, a contradiction. We conclude that L_{2k+2} is not in $\mathcal{HRL}_{2k+1}$. $\square$

5.23 Theorem

The string-graph language $\{(a^n)^* | n \geq 1\}$ is not in $\mathcal{HRL}$.

Proof

The set cannot be generated by a hyperedge-replacement grammar because the growth of the number of edges is not sublinear as required by Corollary 2.6. $\square$

6. Bibliographic Note

The Pumping Lemma (Section 2) is based on the Pumping Lemma for edge-replacement languages given in Kreowski [Kr 79]. A hyperedge-replacement version first was given in [HK 87b]. Moreover, most of the results concerning the generative power (Section 5) are formulated but not proved in [HK 87b].

V. Graph-Theoretic Aspects of HRL's

Graph grammars and graph languages are motivated from a wide spectrum of applications, so that, for the last 20 years, many researchers have felt encouraged to introduce and to investigate an enormous variety of graph-rewriting mechanisms (see the proceedings of the three graph grammar workshops [CER 79], [ENR 83], and [ENRR 87] as a survey on theory and applications). As a class of grammars defines and generates a class of languages, but does not tell much about these languages in general, it is hard to find a class of graph grammars with nice decidability properties and desirable structural results.

At least two approaches are more promising: Boundary node-label-controlled graph grammars [RW 86a+b] and edge-replacement grammars [Co 87a, HK 83+85]. In both cases, the generative mechanisms and powers are of context-free nature, and several — especially graph-theoretic — properties are decidable for the generated graph languages.

The investigation of decision problems concerning graph languages (generated by edge- resp. hyperedge-replacement grammars) is continued and a quite general criterion for decidability is given. Roughly speaking: whenever a graph-theoretic property is “compatible” with the derivation process of hyperedge-replacement grammars in a well-defined manner, then, for each given hyperedge-replacement grammar, the questions

- (1) Is there a graph in the generated language satisfying this property?
- (2) Do all graphs in the generated language satisfy this property?

are decidable.

Beside these questions, we investigate the decidability of a different type of problems concerning functions on graphs and, in particular, numeric quantities like the numbers of nodes, edges, and paths, the node degree, maximum and minimum lengths of paths and cycles, etc. The kind of question we ask for a class of grammars may be called Boundedness Problem. It is as follows:

- (3) Is it decidable whether (or not), concerning a particular quantity, the values of all graphs generated by a grammar are bounded?

For example, we want to know whether the node degree or the number of paths grow beyond any bound within a graph language. As a main result, we show that such a Boundedness Problem is decidable for a class of hyperedge-replacement grammars if the corresponding quantity function is built up by maxima, sums, and products and if the function is compatible with the derivation process of the given grammars. Examples of this kind are the bounded-node-degree problem, the bounded-maximum-path-length problem, the bounded-maximum-number-of-paths problem and others. It should be mentioned that the only result of same nature occurring in the literature is the decidability of the bounded-degree problem for NLC grammars (see [JRW 86]).

The chapter is organized as follows: In Section 1, a theorem fundamental for the following investigations and closely related to the Context-Freeness Lemma (given in Chapter II, Section 2) is presented. In Section 2, graph-theoretic properties like total disconnectedness, connectedness, existence of Eulerian paths and cycles, and k -edge-colorability are

discussed and it is shown that these properties can be easily tested for a graph provided that a derivation of the graph is known. In Section 3, a formal definition of compatibility is given and the graph properties investigated in Section 2 are shown to be compatible. The set of compatible graph properties turns out to be closed under Boolean operations. A metatheorem is stated saying that the compatibility of a graph property yields the decidability of the questions (1) and (2) mentioned above. Sections 4 and 5 are parallel to sections 2 and 3. In Section 4, we discuss several examples of numeric functions which are compatible with the derivation process of our grammars in a certain way. In Section 5, we introduce the general notion of compatible functions, relate them with the notion of compatible predicates, and show that the Boundedness Problem (3) corresponding to a numeric function is decidable if the function is pointwise defined as the maximum of sums and products and if it is compatible.

1. Preliminaries

The results presented in the following sections are mainly based on the Context-Freeness Lemma given in 2.4 of Chapter II. It may be remembered that hyperedge-replacement derivations cannot interfere with each other as long as they handle different hyperedges. On the one hand, a collection of derivations of the form $e^* \xRightarrow{*} H(e)$, for $e \in E_R$, can be simultaneously embedded into a hypergraph R leading to a single derivation $R \xRightarrow{*} H$. On the other hand, restricting a derivation $R \xRightarrow{*} H$ to a handle e^* induced by a hyperedge $e \in E_R$, one obtains a so called "fibre" $e^* \xRightarrow{*} H(e)$ of $R \xRightarrow{*} H$, where $H(e) \subseteq H$. Finally, restricting a derivation to the handles introduced by the hyperedges, and subsequently embedding them again returns the original derivation. In other words, hyperedge-replacement derivations can be distributed to the handles of the hyperedges without losing information. We state and use this result in the following recursive version concerning terminal hypergraphs which are derivable from handles.

1.1 Theorem (decomposition of derivations)

Let $HRG = (N, T, P, Z)$ be a typed and well-formed hyperedge-replacement grammar ¹, $A \in N \cup T$, and $H \in \mathcal{H}_T$. Then there is a derivation $A^* \Rightarrow R \xRightarrow{k} H$ for some $k \geq 0$ if and only if $A^* \Rightarrow R$ and, for each $e \in E_R$, there is a derivation $l_R(e)^* \xRightarrow{k} H(e)$ with $H(e) \subseteq H$ such that $H \cong REPLACE(R, repl)$ with $repl(e) = H(e)$ for $e \in E_R$.

Proof

Theorem 1.1 follows immediately from the Context-Freeness Lemma given in Section 2 of Chapter II: Given a derivation $der : A^* \Rightarrow R \xRightarrow{k} H$ for some $k \geq 0$, then, one may construct the fibres $fibre(e) : l_R(e)^* \xRightarrow{k} H(e)$ of $R \xRightarrow{k} H$. Then, for $e \in E_R$, $H(e) \subseteq H$ and, by the Context-Freeness Lemma, $EMBED(R, fibre) = R \xRightarrow{k} H$ (up to isomorphism). In particular, $H \cong REPLACE(R, repl)$ with $repl(e) = H(e)$ for $e \in E_R$. Conversely, given a derivation $A^* \Rightarrow R$ and derivations $l_R(e)^* \xRightarrow{k} H(e)$ ($e \in E_R$), then the embedding of the derivations into R yields a derivation $R \xRightarrow{k} H'$ with $H' = REPLACE(R, repl)$ and $repl(e) = H(e)$, for $e \in E_R$. If, for some H , $H \cong REPLACE(R, repl)$ and $H(e) \subseteq H$ for $e \in E_R$, then there is a derivation $R \xRightarrow{k} H$. □

¹ For simplicity, we assume that the considered hyperedge-replacement grammars $HRG = (N, T, P, Z)$ are typed w.r.t. $N \cup T$, i.e., there is a function $ltype : N \cup T \rightarrow \mathbb{N} \times \mathbb{N}$ such that, for each production $(A, R) \in P$ and each $e \in E_R$, $ltype(A) = ltype(R)$ and $ltype(l_R(e)) = ltype(e)$. Then, for a symbol $A \in N \cup T$ with $ltype(A) = (m, n)$, A^* denotes an (m, n) -handle induced by A . [Note that (m, n) -handles induced by the same symbol A are isomorphic.] Moreover, it is important to assume that the hyperedge-replacement grammars are well-formed; in this case, for each right-hand side R and each $e \in E_R$, $l_R(e)^* = e^*$ (up to isomorphism).

Remarks

1. The situation may be illustrated as follows.

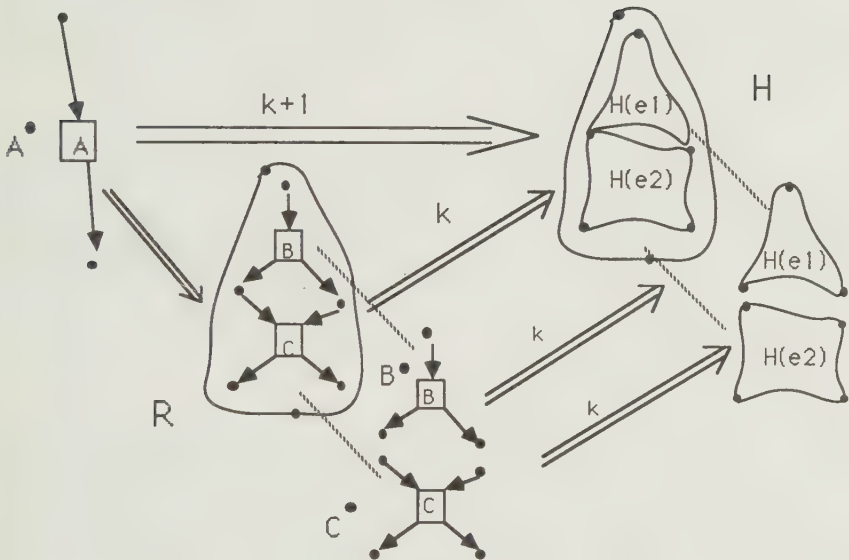


Figure 1.1: Decomposition of a Derivation $A^* \xRightarrow{k+1} H$.

- Given a derivation $R \xRightarrow{k} H$, for each $e \in E_R$, the derivation $l_R(e)^* \xRightarrow{k} H(e)$ is called the *fibre* of e and — the other way round — the given derivation is the *joint embedding* of its fibres.
- Note that the derivation $l_R(e)^* \xRightarrow{k} H(e)$ may be valid or not. In the first case, it has the same form as the original derivation, but it is shorter as the original one. In the latter case, $H(e)$ is isomorphic to e^* (resp. $l_R(e)^*$) and, hence, a terminal handle.

2. Some Compatible Graph-Theoretic Properties

A hyperedge-replacement grammar, as a generating device, specifies a (hyper)graph language. Unfortunately, the generating process produces never more than a finite section of the language explicitly (and even this may consume much time). Hence, one may wonder what the hyperedge-replacement grammar can tell us about the generated language.

As a matter of fact, by Theorem 1.1, we have the following nice situation. Given a hyperedge-replacement grammar and an arbitrary terminal (hyper)graph H with derivation $A^* \Rightarrow R \Rightarrow H$, we get a decomposition of H into “smaller” components which are derivable from the handles of the hyperedges in R . If one is interested in graph-theoretic properties of derived (hyper)graphs, one may ask how a certain property of a derived (hyper)graph depends on properties of the components and of the involved (hyper)graph R . If a property is “compatible” with the derivation process of hyperedge-replacement grammars, i.e., if it can be tested for each derived (hyper)graph H by testing the property (or related properties) for the components and composing the results to a result for H , then a hyperedge-replacement grammar can tell us whether (1) there is a (hyper)graph in the generated language having this property and whether (2) all (hyper)graphs in the generated language have this property.

In this section, we pick up several graph-theoretic properties and show that they are “compatible” with the replacement process of hyperedges. A formal definition of compatibility as well as a metatheorem saying that compatibility implies decidability of the questions (1) and (2) is given in the next section. We are going to have a close look at connectedness, existence of Eulerian paths and cycles, bounded degree, and edge-colorability. To accustom to our kind of investigations, we first consider the question whether a given hypergraph H is *totally disconnected*, i.e., whether $E_H = \emptyset$.

Total Disconnectedness

Given a typed and well-formed hyperedge-replacement grammar $HRG = (N, T, P, Z)$, a hypergraph $H \in \mathcal{H}_T$, and a derivation of the form $A^* \Rightarrow R \Rightarrow H$ in HRG , there is a simple method for testing whether H is totally disconnected or not. The proposed method makes use of the fact that H can be obtained from R by replacing each hyperedge $e \in E_R$ by the result $H(e)$ of its fibre.

2.1 Theorem (total disconnectedness)

Let $HRG = (N, T, P, Z)$ be a typed and well-formed hyperedge-replacement grammar, $A^* \Rightarrow R \Rightarrow H$ a derivation in HRG with $A \in N \cup T$ and $H \in \mathcal{H}_T$, and, for $e \in E_R$, $l_R(e)^* \Rightarrow H(e)$ the fibre of $R \Rightarrow H$ induced by e . Then H is totally disconnected if and only if $H(e)$ is totally disconnected for all $e \in E_R$.

Proof

Without loss of generality, we can assume that $H = REPLACE(R, repl)$ with $repl(e) = H(e) \subseteq H$ for each $e \in E_R$. If H is totally disconnected, i.e., $E_H = \emptyset$, then $E_{H(e)} = \emptyset$ for all $e \in E_R$. Conversely, if $H(e)$ is totally disconnected for all $e \in E_R$, then $E_H = \sum_{e \in E_R} E_{H(e)} = \emptyset$, i.e., H is totally disconnected. $\square$

For simplifying the technicalities, we restrict our following consideration to the class $\mathcal{ERG}$ of edge-replacement grammars in the sense of [HK 83+85]. To be more explicit, a typed and well-formed hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is in $\mathcal{ERG}$ if and only if the right-hand sides of the productions as well as the axiom are (1,1)-graphs. Note that, in this case, each $G \in L(HRG)$ is a (1,1)-graph, i.e., a graph with two distinguished nodes $begin_G$ and end_G .

Connectedness

Now we are going to discuss the question “Is G connected?” for a graph $G \in \mathcal{G}_T$ given by a derivation $A^* \Rightarrow R \xRightarrow{*} G$ in an edge-replacement grammar $ERG = (N, T, P, Z)$ in terms of the fibres $e^* \Rightarrow G(e)$ induced by the edges of R .

Obviously, a graph G is connected whenever R is connected and $G(e)$ is connected for all $e \in E_R$. The converse relationship does not hold as the following example shows.

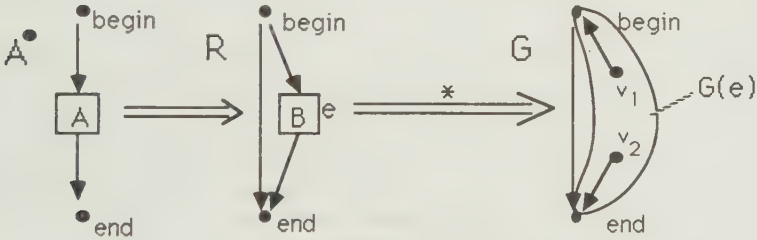


Figure 2.1: A derivation of the form $A^* \Rightarrow R \xRightarrow{*} G$.

Consider the derivation $A^* \Rightarrow R \xRightarrow{*} G$ in which R (consisting of a terminal edge as well as a nonterminal edge e) is connected and the derived graph G (consisting of three terminal edges) is connected, too. The graph $G(e)$ (marked by the “half-moon”) derived from the handle of e is not connected. It is “semi-connected” in the following sense: each node of $G(e)$ is joined with $begin_{G(e)} = begin$ or $end_{G(e)} = end$; addition of an edge between $begin_{G(e)}$ and $end_{G(e)}$ would yield a connected graph. Note that the removal of the edge $e \in E_R$ (the handle of which derives the non-connected graph $G(e)$) from the connected graph R yields a connected graph $R' = R - \{e\}$, again. The connection of the nodes $v_1, v_2 \in V_{G(e)} \subseteq V_G$ is given by three edges, the edge joining the nodes v_1 and $begin_{G(e)} = begin$ in $G(e)$, the edge outside from $G(e)$ connecting $begin$ and end , and the edge joining $end = end_{G(e)}$ and v_2 in $G(e)$.

As this example shows, beside the property “connectedness” the property “semi-connectedness” is needed to characterize “connectedness” of graphs derived from handles. In the following a graph $G \in \mathcal{G}_T$ is said to be *semi-connected* if each node $v \in V_G - \{begin_G, end_G\}$ is joined with $begin_G$ or end_G . It is called *connected* if each node $v \in V_G - \{begin_G, end_G\}$ is joined with $begin_G$ and end_G .

2.2 Theorem (connectedness)

Let $ERG = (N, T, P, Z)$ be an edge-replacement grammar, $A^* \Rightarrow R \xRightarrow{*} G$ a derivation in ERG with $A \in N \cup T$ and $G \in \mathcal{G}_T$, and, for $e \in E_R$, $l_R(e)^* \xRightarrow{*} G(e)$ the fibre of $R \xRightarrow{*} G$ induced by e . Then

- (1) G is connected if and only if $R' = R - \{e \in E_R | G(e) \text{ is not connected}\}$ is connected and $G(e)$ is semi-connected for all $e \in E_R$.
- (2) G is semi-connected if and only if $R' = R - \{e \in E_R | G(e) \text{ is not connected}\}$ is semi-connected and $G(e)$ is semi-connected for all $e \in E_R$.

Proof

Without loss of generality, we may assume that $V_R \subseteq V_G$ and $G = REPLACE(R, repl)$ with $repl(e) = G(e) \subseteq G$ for each $e \in E_R$.

1. Let G be connected. Then $G(e)$ is semi-connected for each $e \in E_R$, which may be seen as follows: If, for some $e \in E_R$, $G(e)$ would not be semi-connected, then there would be a connected component of $G(e)$ neither containing $begin_{G(e)}$ nor $end_{G(e)}$ which would be a connected component in G , too. Moreover, $R' = R - \{e \in E_R | G(e) \text{ is not connected}\}$ turns out to be connected: For $v_1, v_2 \in V_R \subseteq V_G$ there exists a path p in G , which joins v_1 and v_2 .¹ Running through the path, some nodes of R are visited, say $w_0, \dots, w_n$. Cutting p in these nodes, one gets a sequence of paths $p_1, \dots, p_n$ (where p_i connects w_{i-1} and w_i for $i = 1, \dots, n$). For each $i = 1, \dots, n$, there exists $e_i \in E_R$ such that p_i belongs to $G(e_i)$. Obviously, e_i connects w_{i-1} and w_i for $i = 1, \dots, n$. Thus, the edges $e_1, \dots, e_n$ form a path from v_1 to v_2 in R . Note that $e_1, \dots, e_n \in E_{R'}$, because the graphs $G(e_1), \dots, G(e_n)$ are semi-connected as well as begin-end connected and, hence, connected.

Conversely, let $R' = R - \{e \in E_R | G(e) \text{ is not connected}\}$ be connected and $G(e)$ be semi-connected for all $e \in E_R$. Let $v_1, v_2 \in V_G$. Three cases may occur.

Case 1: $v_1, v_2 \in V_R$. Then R has a path from v_1 to v_2 consisting, say, of edges $e_1, \dots, e_n$, and, $G(e_i)$, for $i = 1, \dots, n$ has a path p_i connecting $begin_{G(e_i)}$ and $end_{G(e_i)}$. Therefore, the sequence of paths $p_1, \dots, p_n$ forms a path p from v_1 to v_2 in G .

Case 2: $v_1 \in INT_{repl(e)}$ (for some $e \in E_R$) and $v_2 \in V_R$. By semi-connectedness of $repl(e)$, there is a node $v'_1 \in EXT_{repl(e)}$ such that v_1 and v'_1 are joined by a path in $repl(e)$. Moreover, v'_1 and v_2 are joined by a path in G (cf. case 1). Thus, there is a path from v_1 to v_2 in G .

Case 3: $v_i \in INT_{repl(e_i)}$ for some $e_i \in E_R$ ($i = 1, 2$). Using the same arguments as in case 2, there is a node $v'_i \in EXT_{repl(e_i)}$, such that v_i and v'_i are joined by a path in $repl(e_i)$ ($i = 1, 2$). Moreover, v'_1 and v'_2 are joined by a path in G (cf. case 1). Thus, there is a path from v_1 to v_2 in G . This completes the proof of the first statement.

2. The second statement can be proved in a similar way. □

For expressing the interrelations between the statements concerning "connectedness" and "semi-connectedness" we make use of the predicates $CONN$ and $CONN'$ defined on pairs (G, x) , with $G \in \mathcal{G}_T$ and $x \in \{co, semico\}$, and triples $(R, assign, x)$, with $R \in \mathcal{G}_{N \cup T}$,

¹ A path is meant to be a sequence $p = e_1, e_2, \dots, e_n$ of edges such that, for $1 \leq i < n$, e_i and e_{i+1} are adjacent.

$assign : E_R \rightarrow \{co, semico\}$, and $x \in \{co, semico\}$, respectively:

- $CONN(G, co)$ if and only if G is connected,
 $CONN(G, semico)$ if and only if G is semi-connected,
- $CONN'(R, assign, co)$ if and only if $R' = R - \{e \in E_R \mid assign(e) \neq co\}$ is connected,
 $CONN'(R, assign, semico)$ if and only if $R' = R - \{e \in E_R \mid assign(e) \neq co\}$ is semi-connected.

Now the theorem can be reformulated as follows:

2.3 Corollary (connectedness)

Let $ERG = (N, T, P, Z)$ be an edge-replacement grammar, $A^* \Rightarrow R \Rightarrow G$ a derivation in ERG with $A \in N \cup T$ and $G \in \mathcal{G}_T$, and, for $e \in E_R$, $l_R(e)^* \Rightarrow G(e)$ the fibre of $R \Rightarrow G$ induced by e . Then, for $x \in \{co, semico\}$, $CONN(G, x)$ holds if and only if there is a mapping $assign : E_R \rightarrow \{co, semico\}$ such that $CONN'(R, assign, x)$ as well as $CONN(G(e), assign(e))$ for all $e \in E_R$.

Proof

For $x = co$ (resp. $x = semico$), suppose $CONN(G, x)$ holds. By Theorem 2.2, $R' = R - \{e \in E_R \mid G(e) \text{ is not connected}\}$ is connected (resp. semi-connected) and $G(e)$ is semi-connected for all $e \in E_R$. Now choose $assign : E_R \rightarrow \{co, semico\}$ by $assign(e) \neq co$ iff $G(e)$ is not connected ($e \in E_R$). Then $CONN'(R, assign, x)$ is satisfied. Moreover, $CONN(G(e), assign(e))$ holds for $e \in E_R$ with $assign(e) = semico$ and, by choice of $assign$, $CONN(G(e), assign(e))$ holds for $e \in E_R$ with $assign(e) = co$, too. Conversely, suppose $assign : E_R \rightarrow \{co, semico\}$ is given such that $CONN'(R, assign, co)$ (resp. $CONN'(R, assign, semico)$) and $CONN(G(e), assign(e))$ for $e \in E_R$ hold. Then, $R - \{e \in E_R \mid G(e) \text{ is not connected}\} \supseteq R - \{e \in E_R \mid assign(e) \neq co\}$ is connected (resp. semi-connected) and $G(e)$ is semi-connected for all $e \in E_R$, because each connected graph is also semi-connected. $\square$

Eulerian Graphs

We are going to discuss the question “Is G Eulerian?”, i.e., “Is it possible to find a path that traverses each edge of the graph exactly once, goes through all nodes, and ends at the starting node?”. For this purpose, we use the well-known fact that a graph G is Eulerian if and only if it is connected and every node of G has even degree. The problem whether G is connected (or not) can be handled as in 2.2. Hence it remains to find a criterion for testing whether every node of G has even degree or not.

The “even-degree property” of a graph G with derivation $A^* \Rightarrow R \Rightarrow G$ can be characterized in terms of the fibres $l_R(e)^* \Rightarrow G(e)$. The characterization is based on the following observation. Let R be a graph and v be a node in R . If an edge e outgoing from v is replaced by some graph $G(e)$, then the degree of $begin_{G(e)}$ has to be taken into account whenever the degree of v in G is considered; accordingly, the degree of $end_{G(e)}$ has to be taken in consideration if e is incoming into the node v . Now the degree of v in G can be determined by summing up the degrees of the nodes $begin_{G(e)}$ in $G(e)$ for edges e outgoing from $v \in V_R$ and $end_{G(e)}$ in $G(e)$ for edges e incoming into $v \in V_R$.

Proof

Without loss of generality, we may assume that $V_R \subseteq V_G$ and $G(e) \subseteq G$ for all $e \in E_R$. Suppose $EUL(G, (i, j))$ holds. Then, $d_G(\text{begin}_G) \bmod 2 = i$, $d_G(\text{end}_G) \bmod 2 = j$, and $d_G(v) \bmod 2 = 0$ for $v \in INT_G$. Choosing $\text{assign} : E_R \rightarrow \{0, 1\}^2$ by

$$\text{assign}(e) = (d_{G(e)}(\text{begin}_{G(e)}) \bmod 2, d_{G(e)}(\text{end}_{G(e)}) \bmod 2)$$

for all $e \in E_R$, we obtain $EUL(G(e), \text{assign}(e))$ for all $e \in E_R$. Moreover, for $v \in V_R$,

$$\begin{aligned} \text{ASSIGN}(v) \bmod 2 &= \left(\sum_{e \in E_R^+(v)} \text{assign}(e)_1 + \sum_{e \in E_R^-(v)} \text{assign}(e)_2 \right) \bmod 2 \\ &= \left(\sum_{e \in E_R^+(v)} d_{G(e)}(\text{begin}_{G(e)}) + \sum_{e \in E_R^-(v)} d_{G(e)}(\text{end}_{G(e)}) \right) \bmod 2 \\ &= d_G(v) \bmod 2. \end{aligned}$$

Since, by assumption, $d_G(\text{begin}_G) \bmod 2 = i$, $d_G(\text{end}_G) \bmod 2 = j$, and $d_G(v) \bmod 2 = 0$ for $v \in INT_R$, we obtain $\text{ASSIGN}(\text{begin}_R) \bmod 2 = i$, $\text{ASSIGN}(\text{end}_R) \bmod 2 = j$, and $\text{ASSIGN}(v) \bmod 2 = 0$ for $v \in INT_R$, i.e., $EUL'(R, \text{assign}, (i, j))$ is satisfied. Conversely, assume that, for some $\text{assign} : E_R \rightarrow \{0, 1\}^2$, $EUL(G(e), \text{assign}(e))$ for all $e \in E_R$ as well as $EUL'(R, \text{assign}, (i, j))$ hold. Then, for $v \in V_G$,

$$\begin{aligned} d_G(v) \bmod 2 &= \left(\sum_{e \in E_R^+(v)} d_{G(e)}(\text{begin}_{G(e)}) + \sum_{e \in E_R^-(v)} d_{G(e)}(\text{end}_{G(e)}) \right) \bmod 2 \\ &= \left(\sum_{e \in E_R^+(v)} \text{assign}(e)_1 + \sum_{e \in E_R^-(v)} \text{assign}(e)_2 \right) \bmod 2 \\ &= \text{ASSIGN}(v) \bmod 2. \end{aligned}$$

By assumption, we have $\text{ASSIGN}(\text{begin}_R) \bmod 2 = i$, $\text{ASSIGN}(\text{end}_R) \bmod 2 = j$, and $\text{ASSIGN}(v) \bmod 2 = 0$ ($v \in INT_R$). Thus, $d_G(\text{begin}_G) \bmod 2 = i$, $d_G(\text{end}_G) \bmod 2 = j$, and $d_G(v) \bmod 2 = 0$ for $v \in INT_R$. Consequently, $EUL(G, (i, j))$ is satisfied. $\square$

Remark

With respect to the questions “Is G Eulerian?” resp. “Has G an Eulerian path from begin_G to end_G ?” we get the following relations:

- (1) G has an Eulerian cycle if and only if $\text{CONN}(G, co)$ and $EUL(G, (0, 0))$.
- (2) G has an Eulerian path connecting begin_G and end_G if and only if $\text{CONN}(G, co)$ and $EUL(G, (1, 1))$.⁴

⁴ Notice, that a graph $G \in \mathcal{G}_T$ possesses an Eulerian path from begin_G to end_G if and only if it is connected and begin_G and end_G are the only nodes with odd degree.

Boundedness

In this subsection we will investigate the question “Is G k -bounded?”. Note, that a graph G is said to be k -bounded (for some $k \in \mathbb{N}$) if the degree of each node is bounded by k , i.e., $d_G(v) \leq k$ for all $v \in V_G$.

Similar to the “even-degree property” investigated in 2.4, the “ k -boundedness property” of a graph G with derivation $A^* \Rightarrow R \xrightarrow{*} G$ can be characterized in terms of the fibres $l_R(e)^* \xrightarrow{*} G(e)$. This characterization is based on the observation that k -boundedness of G is induced by k -boundedness of the $G(e)$ provided that, for each $v \in V_R$, the sum of the degrees of the begin-nodes of the $G(e)$ for which e is outgoing from $v \in V_R$ and the end-nodes of the $G(e)$ for which e is incoming into $v \in V_R$ is bounded by k .

Let $G \in \mathcal{G}_T$, $R \in \mathcal{G}_{N \cup T}$, $\text{assign} : E_R \rightarrow [k]^2$, and $i, j \in [k]$ (for some $k \in \mathbb{N}$)⁵. Then

- $k - \text{BOUNDED}(G, (i, j))$ denotes the fact that $\max_{v \in V_G} d_G(v) \leq k$,
 $d_G(\text{begin}_G) = i$, and $d_G(\text{end}_G) = j$.
- $k - \text{BOUNDED}'(R, \text{assign}, (i, j))$ denotes the fact that $\max_{v \in V_R} \text{ASSIGN}(v) \leq k$,
 $\text{ASSIGN}(\text{begin}_R) = i$, and $\text{ASSIGN}(\text{end}_R) = j$
 where, for $v \in V_R$, $\text{ASSIGN}(v) = \sum_{e \in E_R^+(v)} \text{assign}(e)_1 + \sum_{e \in E_R^-(v)} \text{assign}(e)_2$
 with $E_R^+(v) = \{e \in E_R \mid s_R(e) = v\}$ and $E_R^-(v) = \{e \in E_R \mid t_R(e) = v\}$.

2.5 Theorem (boundedness)

Let $ERG = (N, T, P, Z)$ be an edge-replacement grammar, $A^* \Rightarrow R \xrightarrow{*} G$ a derivation in ERG with $A \in N \cup T$ and $G \in \mathcal{G}_T$, and, for $e \in E_R$, $l_R(e)^* \xrightarrow{*} G(e)$ the fibre of $R \xrightarrow{*} G$ induced by e . For $k \in \mathbb{N}$ and $i, j \in [k]$, $k - \text{BOUNDED}(G, (i, j))$ holds if and only if there is a mapping $\text{assign} : E_R \rightarrow [k]^2$ such that $k - \text{BOUNDED}(G(e), \text{assign}(e))$ for all $e \in E_R$ as well as $k - \text{BOUNDED}'(R, \text{assign}, (i, j))$.

Proof

Without loss of generality, we may assume that $V_R \subseteq V_G$ and $G(e) \subseteq G$ for all $e \in E_R$. Suppose $k - \text{BOUNDED}(G, (i, j))$ holds. Then, $\max_{v \in V_G} d_G(v) \leq k$ for $v \in V_R$, i.e., G is k -bounded for $v \in V_R$. Choosing $\text{assign} : E_R \rightarrow [k]^2$ by

$$\text{assign}(e) = (d_{G(e)}(\text{begin}_{G(e)}), d_{G(e)}(\text{end}_{G(e)}))$$

for $e \in E_R$, $k - \text{BOUNDED}(G(e), \text{assign}(e))$ is true for all $e \in E_R$. Moreover, for $v \in V_R$,

$$\begin{aligned} \text{ASSIGN}(v) &= \sum_{e \in E_R^+(v)} \text{assign}(e)_1 + \sum_{e \in E_R^-(v)} \text{assign}(e)_2 \\ &= \sum_{e \in E_R^+(v)} d_{G(e)}(\text{begin}_{G(e)}) + \sum_{e \in E_R^-(v)} d_{G(e)}(\text{end}_{G(e)}) = d_G(v). \end{aligned}$$

By assumption, $\max_{v \in V_G} d_G(v) \leq k$, $d_G(\text{begin}_G) = i$, and $d_G(\text{end}_G) = j$. Thus, we obtain $\max_{v \in V_R} \text{ASSIGN}(v) \leq k$, $\text{ASSIGN}(\text{begin}_R) = i$, and $\text{ASSIGN}(\text{end}_R) = j$, i.e., $k - \text{BOUNDED}'(R, \text{assign}, (i, j))$ is satisfied. Conversely, assume that, for some $\text{assign} : E_R \rightarrow [k]^2$, $k - \text{BOUNDED}(G(e), \text{assign}(e))$ for all $e \in E_R$ as well as

⁵ For $k \in \mathbb{N}$, $[k]$ denotes the set $[k] = \{0, \dots, k\}$.

$k - \text{BOUNDED}'(R, \text{assign}, (i, j))$ hold. Then, $d_G(v) = d_{G(e)}(v) \leq k$ for $v \in \text{INT}_{G(e)}$ and $e \in E_R$ and

$$\begin{aligned} d_G(v) &= \sum_{e \in E_R^+(v)} d_{G(e)}(\text{begin}_{G(e)}) + \sum_{e \in E_R^-(v)} d_{G(e)}(\text{end}_{G(e)}) \\ &= \sum_{e \in E_R^+(v)} \text{assign}(e)_1 + \sum_{e \in E_R^-(v)} \text{assign}(e)_2 = \text{ASSIGN}(v) \end{aligned}$$

for $v \in V_R \subseteq V_G$. Thus,

$$\max_{v \in V_G} d_G(v) = \max(\max_{e \in E_R} \max_{v \in \text{INT}_{G(e)}} d_{G(e)}(v), \max_{v \in V_R} d_G(v)) \leq \max(k, \max_{v \in V_R} \text{ASSIGN}(v)).$$

By assumption, $\max_{v \in V_R} \text{ASSIGN}(v) \leq k$, $\text{ASSIGN}(\text{begin}_R) = i$, and $\text{ASSIGN}(\text{end}_R) = j$. Consequently, $\max_{v \in V_G} d_G(v) \leq k$, $d_G(\text{begin}_G) = i$, and $d_G(\text{end}_G) = j$, i.e., $k - \text{BOUNDED}(G, (i, j))$ is satisfied. $\square$

Edge-Colorability

In this subsection we will consider the problem of coloring the edges of a graph G , such that no two adjacent edges are uniformly colored. Such a distribution of colors is called a (proper) *edge-coloring* of G . Similar to the k -boundedness investigated in 2.5, the k -edge-colorability of a graph G with derivation $A^* \Rightarrow R \Rightarrow G$ can be characterized in terms of the fibres $l_R(e)^* \Rightarrow G(e)$. This characterization is based on the observation that the k -edge-coloring of G is induced by the k -edge-coloring of the $G(e)$ provided that, for each $v \in V_R$, the cardinality of the union of the color sets $\text{COLORS}_{G(e)}(\text{begin}_{G(e)})$ — the set of colors used for the edges in $G(e)$ incident with $\text{begin}_{G(e)}$ — for which e is outgoing from $v \in V_R$ and $\text{COLORS}_{G(e)}(\text{end}_{G(e)})$ — set of colors used for the edges in $G(e)$ incident with $\text{end}_{G(e)}$ — for which e is incoming into $v \in V_R$ is equal to the degree of v in V_G and also is bounded by k .

To recall the notion of edge-colorability, let $G \in \mathcal{G}_T$, $k \in \mathbb{N}$, and K be a set with k elements. A mapping $c_G : E_G \rightarrow K$ is a k -edge-coloring of G if $c_G(e) \neq c_G(e')$ for each pair $e, e' \in E_G$ of adjacent edges. G is said to be k -edge-colorable if there is a k -edge-coloring of G . For $G \in \mathcal{G}_T$, $R \in \mathcal{G}_{N \cup T}$, $\text{assign} : E_R \rightarrow \mathcal{P}(K)^2$,⁶ and $X, Y \subseteq K$,

- $k - \text{COLOR}(G, (X, Y))$ denotes the fact that there is a k -edge-coloring c_G of G such that $\text{COLORS}_G(\text{begin}_G) = X$ and $\text{COLORS}_G(\text{end}_G) = Y$ where, for $v \in V_G$, $\text{COLORS}_G(v) = \bigcup_{e \in E_G(v)} c_G(e)$ and $E_G(v) = \{e \in E_G \mid s_G(e) = v \text{ or } t_G(e) = v\}$.
- $k - \text{COLOR}'(R, \text{assign}, (X, Y))$ refers to the fact that $\# \text{ASSIGN}(v) = \# E_R(v) \leq k$ ⁷ for all $v \in V_R$, $\text{ASSIGN}(\text{begin}_R) = X$, and $\text{ASSIGN}(\text{end}_R) = Y$ where, for $v \in V_R$, $\text{ASSIGN}(v) = \bigcup_{e \in E_R^+(v)} \text{assign}(e)_1 \cup \bigcup_{e \in E_R^-(v)} \text{assign}(e)_2$ with $E_R^+(v) = \{e \in E_R \mid s_R(e) = v\}$ and $E_R^-(v) = \{e \in E_R \mid t_R(e) = v\}$.

⁶ For a set A , $\mathcal{P}(A)$ denotes the power set of A .

⁷ For a set A , $\#A$ denotes the number of elements in A .

2.6 Theorem (edge-colorability)

Let $ERG = (N, T, P, Z)$ be an edge-replacement grammar, $A^* \Rightarrow^* R \Rightarrow^* G$ a derivation in ERG with $A \in N \cup T$ and $G \in \mathcal{G}_T$, and, for $e \in E_R$, $l_R(e)^* \Rightarrow^* G(e)$ the fibre of $R \Rightarrow^* G$ induced by e . For $k \in \mathbb{N}$ and $X, Y \subseteq K$, $k - \text{COLOR}(G, (X, Y))$ holds if and only if there is a mapping $\text{assign} : E_R \rightarrow \wp(K)^2$ such that $k - \text{COLOR}(G(e), \text{assign}(e))$ for all $e \in E_R$ as well as $k - \text{COLOR}'(R, \text{assign}, (X, Y))$.

Proof

Without loss of generality, we may assume that $V_R \subseteq V_G$ and $G(e) \subseteq G$ for all $e \in E_R$. Suppose $k - \text{COLOR}(G, (X, Y))$ holds. Let $c_G : E_G \rightarrow K$ be a k -edge-coloring of G . Because $G(e) \subseteq G$, the restriction of c_G to $E_{G(e)}$ defines an appropriate k -edge-coloring $c_{G(e)}$ of each $G(e)$. Choosing $\text{assign} : E_R \rightarrow \wp(K)^2$ by

$\text{assign}(e) = (\text{COLORS}_{G(e)}(\text{begin}_{G(e)}), \text{COLORS}_{G(e)}(\text{end}_{G(e)}))$ for $e \in E_R$, $k - \text{COLOR}(G(e), \text{assign}(e))$ becomes true for each $e \in E_R$. Furthermore, for $v \in V_R$,

$$\begin{aligned} \text{ASSIGN}(v) &= \bigcup_{e \in E_R^+(v)} \text{assign}(e)_1 + \bigcup_{e \in E_R^-(v)} \text{assign}(e)_2 \\ &= \sum_{e \in E_R^+(v)} \text{COLORS}_{G(e)}(\text{begin}_{G(e)}) + \sum_{e \in E_R^-(v)} \text{COLORS}_{G(e)}(\text{end}_{G(e)}) \\ &= \text{COLORS}_G(v). \end{aligned}$$

By assumption, $\#\text{COLORS}_G(v) \leq k$ for all $v \in V_R$, $\text{COLORS}_G(\text{begin}_G) = X$, and $\text{COLORS}_G(\text{end}_G) = Y$. Thus, $\#\text{ASSIGN}(v) \leq k$ for $v \in V_R$, $\text{ASSIGN}(\text{begin}_R) = X$, and $\text{ASSIGN}(\text{end}_R) = Y$. Moreover, $\#\text{ASSIGN}(v) = \#\text{COLORS}_G(v) = \#E_G(v)$ for $v \in V_R$, because c_G is an edge-coloring of G . Hence, $k - \text{COLOR}(R, \text{assign}, (X, Y))$ holds. Conversely, assume that, for some $\text{assign} : E_R \rightarrow \wp(K)^2$, $k - \text{COLOR}(G(e), \text{assign}(e))$ for $e \in E_R$ as well as $k - \text{COLOR}'(R, \text{assign}, (X, Y))$ hold. Then

$$\begin{aligned} \text{COLORS}_G(v) &= \bigcup_{e \in E_R^+(v)} \text{COLORS}_{G(e)}(\text{begin}_{G(e)}) + \bigcup_{e \in E_R^-(v)} \text{COLORS}_{G(e)}(\text{end}_{G(e)}) \\ &= \bigcup_{e \in E_R^+(v)} \text{assign}(e)_1 + \bigcup_{e \in E_R^-(v)} \text{assign}(e)_2 = \text{ASSIGN}(v). \end{aligned}$$

By assumption, we have $\#\text{ASSIGN}(v) = \#E_G(v) \leq k$ ($v \in V_R$), $\text{ASSIGN}(\text{begin}_R) = X$, and $\text{ASSIGN}(\text{end}_R) = Y$. Consequently, $\#\text{COLORS}_G(v) \leq k$ for $v \in V_R$, $\text{COLORS}_G(\text{begin}_G) = X$, and $\text{COLORS}_G(\text{end}_G) = Y$. Define now $c_G : E_G \rightarrow K$ by $c_G(e') = c_{G(e)}(e')$, for $e' \in E_{G(e)}$ and $e \in E_R$. Then c_G becomes an edge-coloring of G because, for $e \in E_R$, $c_{G(e)}$ is an edge-coloring and, for $v \in V_R \subseteq V_G$, $\#\text{COLORS}_G(v) = \#\text{ASSIGN}(v) = \#E_G(v)$. Hence, $k - \text{COLOR}(G, (X, Y))$ is satisfied. $\square$

3. A General View of Compatible Predicates

All the examples given in Section 2 follow a simple scheme. Roughly speaking, a predicate PR holds for $H \in \mathcal{H}_C$ which is derivable from a handle A^* starting with the production (A, R) if and only if R fulfills some auxiliary predicate PR' and PR holds for all $H(e)$, $e \in E_R$. As the above mentioned example concerning connectedness demonstrates, this view is oversimplified for most applications. To check PR for H , one may have to check some other related properties for the $H(e)$ (with respect to the connectedness of H , some of the $H(e)$ are allowed to be semi-connected). Therefore, we use families of properties indexed by some finite set I and we need a mapping *assign* which determines the property out of the family to be checked for each $H(e)$.

3.1 Definition (compatible-predicates)

1. Let $\mathcal{C}$ be a class of hyperedge-replacement grammars, I a finite index set, $PROP$ a decidable predicate defined on pairs (H, i) , with $H \in \mathcal{H}_C$ and $i \in I$, and $PROP'$ a decidable predicate on triples $(H, assign, i)$, with $H \in \mathcal{H}_C$, a mapping $assign : E_H \rightarrow I$, and $i \in I$.¹ Then $PROP$ is called $\mathcal{C}, PROP'$ -compatible if, for all $HRG = (N, T, P, Z) \in \mathcal{C}$ and all derivations of the form $A^* \Rightarrow R \Rightarrow^* H$, with $A \in N$ and $H \in \mathcal{H}_T$, and for all $i \in I$, the following holds:

$PROP(H, i)$ is true if and only if there is a mapping $assign : E_R \rightarrow I$ such that

$PROP'(R, assign, i)$ is true and

$PROP(H(e), assign(e))$ is true for all $e \in E_R$.

2. A predicate PR on $\mathcal{H}_C$ is called $\mathcal{C}$ -compatible if predicates $PROP$ and $PROP'$ and an index i_0 exist such that $PR = PROP(-, i_0)$ ² and $PROP$ is $\mathcal{C}, PROP'$ -compatible.

Remark

Intuitively, a property is compatible if it can be tested for a large hypergraph with a long fibre by checking the smaller components of the corresponding shorter fibres. Such a property must be closed under isomorphisms because the derivability of hypergraphs is independent of the representation of nodes and hyperedges.

Various explicit examples of compatible predicates were discussed in Section 2.

3.2 Examples (compatible predicates)

1. Let, for $H \in \mathcal{H}_C$, $DIS(H)$ denote the fact that H is totally disconnected. Then DIS is $\mathcal{C}$ -compatible for the class $\mathcal{C}$ of all hyperedge-replacement grammars: Take a one-element set $I = \{*\}$ and the predicates $DISCO$ and $DISCO'$ which are given by

$DISCO(H, *)$ if and only if $DIS(H)$ for all $H \in \mathcal{H}_C$,

$DISCO'(H, assign, *) = \text{true}$ for all $H \in \mathcal{H}_C$.

¹ In the following, we assume that all considered predicates are *closed under isomorphisms*, i.e., if a predicate Φ holds for $H \in \mathcal{H}_C$ and $H \cong H'$, then Φ holds for H' , too.

² For $i \in I$, $PROP(-, i)$ denotes the unary predicate defined by $PROP(-, i)(H) = PROP(H, i)$ for all $H \in \mathcal{H}_C$.

By Theorem 2.1, $DISCO$ is $\mathcal{C}, DISCO'$ -compatible and $DIS = DISCO(-, *)$ is $\mathcal{C}$ -compatible.

2. By Theorem 2.2, the predicates CO and $SEMICO$ given for all $G \in \mathcal{G}_C$ by
 $CO(G)$ if and only if G is connected, i.e. $CONN(G, co)$, and
 $SEMICO(G)$ if and only if G is semi-connected, i.e., $CONN(G, semico)$,
are $\mathcal{C}$ -compatible.

3. By Theorem 2.4, the predicates EUL_C and EUL_P given for all $G \in \mathcal{G}_C$ by
 $EUL_C(G)$ if and only if every node of G has even degree, i.e., $EUL(G, (0, 0))$, and
 $EUL_P(G)$ if and only if every node of G has even degree, except $begin_G$ and end_G ,
i.e., $EUL(G, (1, 1))$

are $\mathcal{C}$ -compatible. Moreover, the predicates $EULER_C$ and $EULER_P$ given by
 $EULER_C(G)$ if and only if $CO(G) \wedge EUL_C(G)$ and
 $EULER_P(G)$ if and only if $CO(G) \wedge EUL_P(G)$

are $\mathcal{C}$ -compatible, provided that the conjunction of $\mathcal{C}$ -compatible predicates is $\mathcal{C}$ -compatible.

4. For $k \in \mathbb{N}$, the predicate $k - BOUND$ given for all $G \in \mathcal{G}_C$ by
 $k - BOUND(G)$ if and only if G is k -bounded,

$$\text{i.e., } \bigvee_{0 \leq i, j \leq k} k - BOUNDED(G, (i, j)),$$

is $\mathcal{C}$ -compatible, provided that the disjunction of $\mathcal{C}$ -compatible predicates is $\mathcal{C}$ -compatible.

5. For $k \in \mathbb{N}$, the predicate $k - COL$ given for all $G \in \mathcal{G}_C$ by

$$k - COL(G) \text{ if and only if } G \text{ has a } k\text{-edge-coloring,}$$

$$\text{i.e., } \bigvee_{X, Y \subseteq K} k - COLOR(G, (X, Y)),$$

is $\mathcal{C}$ -compatible, provided that the disjunction of $\mathcal{C}$ -compatible predicates is $\mathcal{C}$ -compatible.

Closure under Boolean Operations

The examples in 3.2 show that the question of closurenens of $\mathcal{C}$ -compatibility is very important. In fact, $\mathcal{C}$ -compatibility is closed under Boolean operations (cf. Theorem 3.3).

3.3 Theorem (closure under Boolean operations)

Let $\mathcal{C}$ be a class of hyperedge-replacement grammars and PR_1, PR_2 be $\mathcal{C}$ -compatible predicates. Let, for $H \in \mathcal{H}_C$,

$$(PR_1 \wedge PR_2)(H) \text{ if and only if } PR_1(H) \wedge PR_2(H),$$

$$(PR_1 \vee PR_2)(H) \text{ if and only if } PR_1(H) \vee PR_2(H),$$

$$(\neg PR_1)(H) \text{ if and only if } \neg PR_1(H).$$

Then $(PR_1 \wedge PR_2)$, $(PR_1 \vee PR_2)$, and $(\neg PR_1)$ are $\mathcal{C}$ -compatible.

Proof

Let, for $j = 1, 2$, PR_j be a $\mathcal{C}$ -compatible predicate, $PROP_j, PROP'_j$ the corresponding predicates, I_j the index set, and i_j the index such that $PR_j = PROP_j(-, i_j)$.

1. For proving the $\mathcal{C}$ -compatibility of $(PR_1 \wedge PR_2)$, we define new predicates $PROP$ and $PROP'$ derived from the given ones and show the $\mathcal{C}, PROP'$ -compatibility of $PROP$ using the compatibility of the old ones. Let

$$I = I_1 \times I_2,$$

$$PROP(H, (j_1, j_2)) \iff PROP_1(H, j_1) \wedge PROP_2(H, j_2), \text{ and}$$

$$PROP'(H, assign, (j_1, j_2)) \iff PROP'_1(H, assign_1, j_1) \wedge PROP'_2(H, assign_2, j_2)$$

where $assign_1, assign_2$ are the projections of $assign$ to the first and the second component, respectively.

$$\text{Then, for a hypergraph } H \in \mathcal{H}_C, (PR_1 \wedge PR_2)(H) \iff PR_1(H) \wedge PR_2(H)$$

$$\iff PROP_1(H, i_1) \wedge PROP_2(H, i_2) \iff PROP(H, (i_1, i_2)),$$

i.e., $(PR_1 \wedge PR_2) = PROP(-, (i_1, i_2))$. The $\mathcal{C}, PROP'$ -compatibility of $PROP$, can be proved by the following reasoning for a derivation $A^* \Rightarrow R \Rightarrow H$ with $A \in N$ and $H \in \mathcal{H}_T$. By definition of $PROP$, $\mathcal{C}, PROP'_1$ -compatibility of $PROP_1$, and $\mathcal{C}, PROP'_2$ -compatibility of $PROP_2$, we obtain

$$PROP(H, (j_1, j_2))$$

$$\iff PROP_1(H, j_1) \wedge PROP_2(H, j_2)$$

$$\iff (A) \exists assign_1 : E_R \rightarrow I_1, \exists assign_2 : E_R \rightarrow I_2 :$$

$$PROP'_1(R, assign_1, j_1) \wedge PROP'_2(R, assign_2, j_2)$$

$$\wedge \bigwedge_{e \in E_R} PROP_1(H(e), assign_1(e)) \wedge \bigwedge_{e \in E_R} PROP_2(H(e), assign_2(e)).$$

Choosing $assign : E_R \rightarrow I$ as $assign(e) = (assign_1(e), assign_2(e))$ for all $e \in E_R$, $PROP(H(e), assign(e))$ for $e \in E_R$ as well as $PROP'(R, assign, (j_1, j_2))$ become true.

Conversely, if, for some mapping $assign : E_R \rightarrow I$, $PROP(H(e), assign(e))$ (for $e \in E_R$) as well as $PROP'(R, assign, (j_1, j_2))$ hold, then we obtain (A) choosing $assign_1$ and $assign_2$ as the projections of $assign$ to the first and the second component, respectively. Hence $PROP$ is $\mathcal{C}, PROP'$ -compatible and $(PR_1 \wedge PR_2) = PROP(-, (i_1, i_2))$ is $\mathcal{C}$ -compatible.

2. To derive the $\mathcal{C}$ -compatibility of $(\neg PR_1)$, we define — similar to 1. — new predicates

$PROP$ and $PROP'$ and show the $\mathcal{C}, PROP'$ -compatibility of $PROP$. Let

I be the power set of I_1 ,

$$PROP(H, X)$$

$$\iff \bigwedge_{x \in X} PROP_1(H, x) \wedge \bigwedge_{x \notin X} \neg PROP_1(H, x),$$

$$PROP'(H, assign, X)$$

$$\iff \bigwedge_{x \in X} \bigvee_{a \in M(assign)} PROP'_1(H, a, x) \wedge \bigwedge_{x \notin X} \bigwedge_{a \in M(assign)} \neg PROP'_1(H, a, x)$$

$$\text{where } M(assign) = \{a : E_R \rightarrow I_1 \mid a(e) \in assign(e) \text{ for all } e \in E_R\}.$$

Using the definition of $PROP$ and $\mathcal{C}, PROP'_1$ -compatibility of $PROP_1$, we get

$$PROP(H, X) \iff \bigwedge_{x \in X} PROP_1(H, x) \wedge \bigwedge_{x \notin X} \neg PROP_1(H, x)$$

$$\iff \bigwedge_{x \in X} [\bigvee_{a : E_R \rightarrow I_1} PROP'_1(R, a, x) \wedge \bigwedge_{e \in E_R} PROP_1(H(e), a(e))] \wedge \bigwedge_{x \notin X} [\bigvee_{a : E_R \rightarrow I_1} PROP'_1(R, a, x) \wedge \bigwedge_{e \in E_R} \neg PROP_1(H(e), a(e))].$$

Choosing $assign : E_R \rightarrow I$ by $assign(e) = \{i \in I_1 \mid PROP_1(H(e), i)\}$ for all $e \in E_R$, $PROP(H(e), assign(e))$ becomes true for all $e \in E_R$. Moreover, for each $x \in X$, there is a mapping $a : E_R \rightarrow I_1$ such that $PROP'(R, a, x)$ and $a(e) \in assign(e)$ for all $e \in E_R$. For $x \notin X$, there is no mapping $a : E_R \rightarrow I_1$ such that $PROP'_1(R, a, x)$ and $a(e) \in assign(e)$ for all $e \in E_R$. Thus, $PROP'_1(R, assign, X)$ is satisfied.

Conversely, if $PROP'(R, assign, X)$ and $PROP(H(e), assign(e))$ hold for some mapping $assign : E_R \rightarrow I$, then

$$\begin{aligned}
 (B) \quad & \bigwedge_{x \in X} \bigvee_{a \in M(assign)} PROP'_1(R, a, x) \wedge \bigwedge_{e \in E_R} \bigwedge_{x \in assign(e)} PROP_1(H(e), x) \\
 & \Rightarrow \bigwedge_{x \in X} [\bigvee_{a \in M(assign)} PROP'_1(R, a, x) \wedge \bigwedge_{e \in E_R} PROP_1(H(e), a(e))] \\
 & \Rightarrow \bigwedge_{x \in X} PROP_1(H, x)
 \end{aligned}$$

$$\begin{aligned}
 (C) \quad & \bigwedge_{x \notin X} \bigwedge_{a \in M(assign)} \neg PROP'_1(R, a, x) \\
 & \Rightarrow \bigwedge_{x \notin X} [\bigwedge_{a \in M(assign)} \neg PROP'_1(R, a, x) \vee \bigvee_{e \in E_R} \neg PROP_1(H(e), a(e))] \\
 & \Rightarrow \bigwedge_{x \notin X} \neg [\bigvee_{a \in M(assign)} PROP'_1(R, a, x) \wedge \bigwedge_{e \in E_R} PROP_1(H(e), a(e))] \\
 & \Rightarrow \bigwedge_{x \notin X} \neg PROP_1(H, x).
 \end{aligned}$$

By (B) and (C), $PROP(H, X)$ becomes satisfied. This completes the proof of the $\mathcal{C}, PROP'$ -compatibility of $PROP$.

Finally, we will show how the $\mathcal{C}, PROP'$ -compatibility of $PROP$ induces the $\mathcal{C}$ -compatibility of $(\neg PR_1)$. Obviously,

$$\begin{aligned}
 (\neg PR_1)(H) & \iff \neg PR_1(H) \iff \neg PROP_1(H, i_1) \\
 & \iff \neg \bigvee_{i_1 \in S \subseteq I} [\bigwedge_{x \in S} PROP_1(H, x) \wedge \bigwedge_{x \notin S} \neg PROP_1(H, x)] \\
 & \iff \neg \bigvee_{i_1 \in S \subseteq I} PROP(H, S) \iff \bigwedge_{i_1 \in S \subseteq I} \neg PROP(H, S) \\
 & \iff \bigwedge_{i_1 \in S \subseteq I} PROP(H, I - S) \iff \bigwedge_{S \subseteq I \text{ with } i_1 \notin S} PROP(H, S).
 \end{aligned}$$

$\mathcal{C}, PROP'$ -compatibility of $PROP$ implies $\mathcal{C}$ -compatibility of $PR_S = PROP(-, S)$ for all $S \subseteq I$. Since the conjunction of $\mathcal{C}$ -compatible predicates yields a $\mathcal{C}$ -compatible predicate, $\neg PR_1$ becomes $\mathcal{C}$ -compatible.

3. By $(PR_1 \vee PR_2) = \neg(\neg PR_1 \wedge \neg PR_2)$, the $\mathcal{C}$ -compatibility of $(PR_1 \vee PR_2)$ follows from the $\mathcal{C}$ -compatibility of PR_1 , the $\mathcal{C}$ -compatibility of PR_2 , and the fact that the conjunction and negation of $\mathcal{C}$ -compatible predicates yields $\mathcal{C}$ -compatible predicates (see 1. and 2.). $\square$

3.4 Examples (compatible predicates)

By 3.3, the predicates $EULER_C$, $EULER_P$, $k - BOUND$, and $k - COL$ given by

$$\begin{aligned} EULER_C(G) &\text{ if and only if } CO(G) \wedge EUL_C(G), \\ EULER_P(G) &\text{ if and only if } CO(G) \wedge EUL_P(G), \\ k - BOUND(G) &\text{ if and only if } \bigvee_{0 \leq i, j \leq k} k - BOUNDED(G, (i, j)) \\ k - COL(G) &\text{ if and only if } \bigvee_{X, Y \subseteq K} k - COLOR(G, (X, Y)) \end{aligned}$$

are $\mathcal{C}$ -compatible.

Given a $\mathcal{C}$, $PROP'$ -compatible predicate $PROP$, some (hyper)graphs H may possess several properties, i.e., there may exist indices i, j with $i \neq j$ such that $PROP(H, i)$ as well as $PROP(H, j)$ holds. E.g., a $(1,1)$ -graph may be semi-connected as well as connected. On the other hand, there may be hypergraphs possessing none of the properties in consideration. E.g., in Corollary 2.3, the properties "connected" and "semi-connected" are relevant; but there are $(1,1)$ -graphs which are neither connected nor semi-connected. Using methods of the proof of Theorem 3.3, it can be shown that each compatible predicate can be expressed by a predicate $PROP_u$ for which, for each hypergraph H , there is uniquely one index i such that $PROP_u(H, i)$ holds.

3.5 Theorem (compatible predicates)

Let PR be $\mathcal{C}$ -compatible for some class $\mathcal{C}$ of hyperedge-replacement grammars. Then there are an index set I_u , and predicates $PROP_u$, $PROP'_u$ such that

- $PR = \bigvee_{i \in I_u} PROP_u(-, i)$ for some set $I_0 \subseteq I_u$
- $PROP_u$ is $\mathcal{C}$, $PROP'_u$ -compatible,
- $\forall H \in \mathcal{H}_C \exists ! i \in I_u : PROP_u(H, i)$
- $\forall R \in \mathcal{H}_C \forall assign : E_R \rightarrow I_u \exists ! i \in I_u : PROP'_u(R, assign, i)$.

Proof

Let PR be $\mathcal{C}$ -compatible. Then, by definition, there exist predicates $PROP$, $PROP'$, an index set I , and an index $i_0 \in I$, such that $PROP$ is $\mathcal{C}$, $PROP'$ -compatible and $PROP = PROP(-, i_0)$. Preceding as in the proof of Theorem 3.3, I_u is defined to be the power set of I ,

$$PROP_u(H, X)$$

$$\iff \bigwedge_{x \in X} PROP(H, x) \wedge \bigwedge_{x \notin X} \neg PROP(H, x),$$

$$PROP'_u(H, assign, X)$$

$$\iff \bigwedge_{x \in X} \bigvee_{a \in M(assign)} PROP'(H, a, x) \wedge \bigwedge_{x \notin X} \bigwedge_{a \in M(assign)} \neg PROP'(H, a, x)$$

$$\text{where } M(assign) = \{a : E_R \rightarrow I \mid a(e) \in assign(e) \text{ for all } e \in E_R\}.$$

Moreover, let $I_0 = \{X \in I_u \mid i_0 \in X\}$.

Then $PROP_u$ is $\mathcal{C}$, $PROP'_u$ -compatible and $PR = PROP(-, i_0) = \bigvee_{X \in I_0} PROP_u(-, X)$. It remains to show that, for all H , all R , and all $assign$, there exists uniquely one $i \in I_u$ such that $PROP_u(H, i)$ resp. $PROP'_u(R, assign, i)$ is satisfied. For a hypergraph $H \in \mathcal{H}_C$, choose $X = \{i \in I \mid PROP(H, i) \text{ is satisfied}\}$. Then, by definition of

$PROP_u$, $PROP_u(H, X)$ is satisfied. If $PROP_u(H, X) = PROP_u(H, X')$, then, by definition of $PROP_u$, $X = X'$. For $R \in \mathcal{H}_C$ and a mapping $assign : E_R \rightarrow I_u$, let $X = \{i \in I \mid PROP'(R, a, i) \text{ for some } a \in M(assign)\}$. Then, by definition of $PROP'_u$, $PROP'_u(R, assign, X)$ is satisfied and if $PROP'_u(R, assign, X) = PROP'_u(R, assign, X')$, then $X = X'$. This completes the proof. $\square$

A Linear-Time Algorithm

In this subsection we consider compatible hypergraph properties PR and show that whenever a hypergraph $H \in L(HRG)$ and a derivation (resp. a derivation tree T) of H is known, then we can decide $PR(H)$ in linear time with respect to the size of T .

3.6 Theorem (linear time)

Let PR be $\mathcal{C}$ -compatible with respect to some class $\mathcal{C}$ of hyperedge-replacement grammars. Then, for all $HRG \in \mathcal{C}$, all $H \in L(HRG)$, and all derivation trees T of H , it can be decided in $O(|T|)$ time whether $PR(H)$ holds. [The constant factor depends on HRG and PR .]

Proof

Let PR be $\mathcal{C}$ -compatible for some class $\mathcal{C}$ of hyperedge-replacement grammars. Then, by Theorem 3.5, there are an index set I , and predicates $PROP$, $PROP'$ satisfying the conditions in Theorem 3.5. Given a hypergraph H and a derivation tree T of H (for some $HRG \in \mathcal{C}$), process in a bottom-up fashion.

(1) For each leaf l of T representing the hypergraph L , determine the index i for which $PROP(L, i)$ holds. [By choice of $PROP$, there exists exactly one index with this property]
 (2) For a node v with R -labeled outgoing hyperedge and direct descendants $v_1, \dots, v_m$, the index of v is determined from the indices of $v_1, \dots, v_m$, $i_1, \dots, i_m$, as follows: Choose $assign : E_R \rightarrow I$ by $assign(e_j) = i_j$ ($j = 1, \dots, m$). and determine the index $i \in I$ for which $PROP'(R, assign, i)$ holds. [By choice of $PROP'$, there exists exactly one index with this property.]

This proceeding yields an index for each node. By $\mathcal{C}, PROP'$ -compatibility of $PROP$, the index of a node v in the derivation tree T is i if and only if $PROP(H(v), i)$ holds (where $H(v)$ denotes the hypergraph represented by the subtree with root v). This may be seen as follows. For leaves, the claim follows immediately from (1). Assume now that the statement holds for the nodes $v_1, \dots, v_m$. Let $H(e_1), \dots, H(e_m)$ be the hypergraphs represented by the nodes $v_1, \dots, v_m$, and $i_1, \dots, i_m$ are the corresponding indices. Then $PROP(H(e_1), i_1), \dots, PROP(H(e_m), i_m)$ hold. Let $assign : E_R \rightarrow I$ as given above. By choice of i , $PROP'(R, assign, i)$ is satisfied. By $\mathcal{C}, PROP'$ -compatibility of $PROP$, $PROP(H, i)$ with $H = REPLACE(R, repl)$ and $repl(e_j) = H(e_j)$ (for $j = 1, \dots, m$) is satisfied. On the other hand, H is the hypergraph represented by the node v .

Let now H be the hypergraph represented by T and i the index of the root of T . Then $PROP(H, i)$ is satisfied. On the other hand, $PR(H)$ holds if and only if $\bigvee_{j \in I_0} PROP(H, j)$. Therefore, $PR(H)$ holds if and only if $i \in I_0$.

To see, that the proceeding takes linear time let us mention that, for each hyperedge-replacement grammar HRG , the size of the handles represented by leaves of T as well as the size of the right-hand sides is bounded by a constant depending on HRG and the index

set I is finite. Therefore, each procedure call “DETERMINE INDEX” takes constant time and “DETERMINE INDEX” is called once for each node of the derivation tree, the cost to determine the index of the root is $O(n)$, where n is the number of nodes in the tree. $\square$

A Metatheorem for Graph-Theoretic Decision Problems

Given a (hyper)graph property PR and a class $\mathcal{C}$ of hyperedge-replacement grammars, we are going to study two types of questions for all $HRG \in \mathcal{C}$: “Does PR hold for some $H \in L(HRG)$?” and “Does PR hold for all $H \in L(HRG)$?”. Both questions turn out to be decidable, provided that PR is $\mathcal{C}$ -compatible. We call this result “metatheorem” because of its generic character: Whenever one can prove the compatibility of a property (and we have given various examples in the previous section), one gets a particular decision result for this property as corollary of the metatheorem.

3.7 Theorem (metatheorem for decision problems)

Let PR be $\mathcal{C}$ -compatible with respect to some class $\mathcal{C}$ of hyperedge-replacement grammars. Then, for all $HRG \in \mathcal{C}$, it is decidable whether

- (1) PR holds for some $H \in L(HRG)$,
- (2) PR holds for all $H \in L(HRG)$.

Proof

1. Let PR be $\mathcal{C}$ -compatible for some class $\mathcal{C}$ of hyperedge-replacement grammars and $PROP$ and $PROP'$ the corresponding predicates over the index set I such that $PROP$ is $\mathcal{C}, PROP'$ -compatible and $PR = PROP(-, i_0)$ for some $i_0 \in I$.

Let $HRG = (N, T, P, Z) \in \mathcal{C}$. Without loss of generality, we may assume that HRG is proper. For simplicity, we additionally assume that HRG is typed with respect to $N \cup T$. (If this condition is not satisfied, we consider the completely typed version of HRG .) Then the main task of the proof is to construct the sets

$$V_{PROP, i} = \{A \in N \cup T \mid \text{there is an } H \in \mathcal{H}_C \text{ such that } A^* \xrightarrow{P} H \text{ and } PROP(H, i)\}$$

for all $i \in I$. Then the decision problem reduces to checking whether $l(Z) \in V_{PROP, i_0}$.

For constructing the set $V_{PROP, i}$, we define sets $V_{PROP, i, k}$ for $k \in \mathbb{N}$ recursively as follows:

$$V_{PROP, i, 0} = \{A \in T \mid PROP(A^*, i) \text{ is satisfied}\},$$

$$V_{PROP, i, k+1} = V_{PROP, i, k} \cup \{A \in N \mid (A, R) \in P \wedge \exists assign : E_R \rightarrow I :$$

$$PROP'(R, assign, i) \text{ and } l_R(e) \in V_{PROP, i, k} \text{ for all } e \in E_R\}.$$

Since $V_{PROP, i, k} \subseteq V_{PROP, i, k+1} \subseteq N \cup T$ for all $i \in I$ and $k \in \mathbb{N}$ and $N \cup T$ is finite, there is some $l \in \mathbb{N}$ such that $V_{PROP, i, l} = V_{PROP, i, l+1}$. Obviously, all $V_{PROP, i, k}$ up to the smallest possible l can be constructed effectively. Hence, it remains to prove that $V_{PROP, i} = V_{PROP, i, l}$, which is an immediately consequence of the following two claims.

Claim 1: $V_{PROP, i, k} \subseteq V_{PROP, i}$ for all $i \in I$ and $k \in \mathbb{N}$.

Proof of Claim 1 (by induction on k): The statement is trivially true for $k = 0$. Assume that it is true for k , and let A be in $V_{PROP, i, k+1}$. If A is also in $V_{PROP, i, k}$, the inductive step is trivial. If A is in $V_{PROP, i, k+1} - V_{PROP, i, k}$, then there is a production $(A, R) \in P$ such that $l_R(e) \in V_{PROP, i, k}$ for all $e \in E_R$ and $PROP'(R, assign, i)$ for some mapping $assign : E_R \rightarrow I$. By inductive hypothesis, $l_R(e) \in V_{PROP, i}$ for all $e \in E_R$. Therefore,

there is a derivation $l_R(e)^* \xRightarrow{*} H(e)$ for some $H(e) \in \mathcal{H}_T$ with $PROP(H(e), assign(e))$. Joint embedding of these derivations into R yields a derivation $R \xRightarrow{*} H$ where $H \in \mathcal{H}_T$. Moreover, there is a direct derivation $A^* \Rightarrow R$ through the production (A, R) . Composing these derivations, we get a derivation $A^* \xRightarrow{*} H$ where $H \in \mathcal{H}_T$. Since $PROP$ is $\mathcal{C}$, $PROP'$ -compatible, $PROP(H, i)$ is satisfied. Altogether, it turns out that $A \in V_{PROP, i}$. This proves Claim 1.

Claim 2: $V_{PROP, i} \subseteq V_{PROP, i+1}$.

Proof of Claim 2: $A \in V_{PROP, i}$ means that there is a derivation $A^* \xRightarrow{*} H$ with $H \in \mathcal{H}_T$ and $PROP(H, i)$. Thus, we prove the statement

If $A \in N \cup T$ and $A^* \xRightarrow{*} H$ for some $H \in \mathcal{H}_T$ with $PROP(H, i)$, then $A \in V_{PROP, i+1}$ by induction on the number of derivation steps. The basis, $k = 0$, is trivial; for each zero-step derivation $A^* \xRightarrow{0} H$, $H \cong A^*$, but $A^* \notin \mathcal{H}_T$. Assume that the statement is true for k -step derivations, and let $A^* \xRightarrow{k+1} H$ with $H \in \mathcal{H}_T$ and $PROP(H, i)$. Then the derivation decomposes into $A^* \Rightarrow R' \xRightarrow{k} H$ for some $R' \in \mathcal{H}_{N \cup T}$. If the first derivation step is a dummy, then $A^* \cong R'$, and $R' \xRightarrow{k} H$ can be modified into $A^* \xRightarrow{k} H$. By inductive hypothesis, this implies $A \in V_{PROP, i+1}$. Otherwise, $R' \cong R$ for some production $(A, R) \in P$. Consider now the fibres $l_{R'}(e)^* \xRightarrow{*} H(e)$ of $R \xRightarrow{*} H$ for $e \in E_{R'}$. By $\mathcal{C}$, $PROP'$ -compatibility of $PROP$, there is some mapping $assign : E_R \rightarrow I$ such that $PROP'(R, assign, i)$ and $PROP(H(e), assign(e))$ ($e \in E_R$) are satisfied. Using the inductive hypothesis, we get $l_{R'}(e) \in V_{PROP, i+1}$ for $e \in E_{R'}$. Since $R' \cong R$, we obtain $l_R(e) \in V_{PROP, i+1}$ for $e \in E_R$. By definition and choice of l , we get $A \in V_{PROP, i+1} = V_{PROP, i+1}$. This proves Claim 2.

2. Obviously, PR holds for all $H \in L(HRG)$ if and only if there is no $H \in L(HRG)$ such that $\neg PR$ holds. By Theorem 3.3, $\neg PR$ is $\mathcal{C}$ -compatible. Thus, the question whether $\neg PR$ holds for some $H \in L(HRG)$ is decidable. Hence, the decision problem whether PR holds for all $H \in L(HRG)$ can be reduced to the decision problem whether $\neg PR$ holds for some $H \in L(HRG)$. $\square$

Combining the compatibility results of Section 2 with Theorem 3.7 one obtains a list of decidability results.

3.8 Corollary (special decidable problems)

The question: “Does a given edge-replacement grammar generate some or only graphs with property PR ?” is decidable for all edge-replacement grammars and all properties in the following list:

- (1) A graph is totally disconnected.
- (2) A graph is connected (resp. semi-connected).
- (3) All nodes of a graph have even degree.
- (4) A graph has an Eulerian cycle (resp. an Eulerian path).
- (5) The degree of a graph is bounded by k (for some $k \in \mathbb{N}$).
- (6) A graph has an edge-coloring with at most k colors (for some $k \in \mathbb{N}$).

Remarks

1. Remember that *DIS* is compatible for arbitrary hyperedge-replacement grammars.
2. Although in most cases we have restricted to the class $\mathcal{ERG}$ of all edge-replacement grammars, we are convinced that our statements hold even if we consider the class of all hyperedge-replacement grammars which generate ordinary graph languages and use nonterminal hyperedges with a bounded number of tentacles. We even guess that the considered properties are compatible for arbitrary hyperedge-replacement grammars (with a bounded number of tentacles) if their definition is properly adapted to hypergraphs (which seems easy with respect to connectedness, degree properties as well as k -edge-colorability).
3. As shown in [HKV 87], many other properties — like the existence of Hamiltonian paths and cycles, vertex coloring, and subcontraction — are compatible and the corresponding questions are decidable.

A Metatheorem for Intersection Problems

We continue the investigations of basic properties of hyperedge-replacement grammars and languages, where the central question is the following: “If L is a hyperedge-replacement language and PR a graph-theoretic property, is the set of all hypergraphs from L satisfying PR again a hyperedge-replacement language?”

We demonstrate that the class of hyperedge-replacement languages is very “stable” in the sense that for all compatible properties the resulting languages are hyperedge-replacement languages. In particular, the above question gets an affirmative answer, if the property PR is: being connected, Eulerian, Hamiltonian, planar, resp. k -colorable.

3.9 Theorem (metatheorem for intersection problems)

Let PR be a $\mathcal{C}$ -compatible predicate for some class $\mathcal{C}$ of hyperedge-replacement grammars. For every $HRG \in \mathcal{C}$, there is a hyperedge-replacement grammar HRG_{PR} such that

$$L(HRG_{PR}) = \{H \in L(HRG) \mid H \text{ satisfies } PR\}.$$

Proof

Let PR be $\mathcal{C}$ -compatible and $PROP$ and $PROP'$ be the corresponding predicates over the index set I such that $PROP$ is $\mathcal{C}$, $PROP'$ -compatible and $PR = PROP(-, \mathfrak{i}_0)$ for some $\mathfrak{i}_0 \in I$. Let $HRG = (N, T, P, Z) \in \mathcal{C}$. Then, we construct a hyperedge-replacement grammar $HRG' = (N', T, P', Z')$ generating $\{H \in L(HRG) \mid H \text{ satisfies } PR\}$ as follows. Let

- $N' = N \times I$, the elements of which are written A_i rather than $(A, \mathfrak{i})$,
- $P' = \{(A_i, R_{assign})(A, R) \in P, assign : E_R \rightarrow I, \mathfrak{i} \in I, PROP'(R, assign, \mathfrak{i}),$

$$PROP(l_R(e)^*, assign(e)) \text{ for all terminal } e \in E_R\}$$

where R_{assign} , for $R \in \mathcal{H}_{\mathcal{C}}$ and $assign : E_R \rightarrow I$, is obtained from R by relabeling the nonterminal hyperedges according to $assign$, i.e.,

$R_{assign} = (V_R, E_R, s_R, t_R, l, begin_R, end_R)$ with $l(e) = l_R(e)_{assign(e)}$ if $l_R(e) \in N$ and $l(e) = l_R(e)$ otherwise,

- $Z' = Z_{assign}$ where $assign(e(Z)) = i_0$, this means that the label of the only hyperedge of Z is indexed by i_0 .

To show $L(HRG') = \{H \in L(HRG) \mid H \text{ satisfies } PR\}$, we prove Claim 1 and Claim 2.

Claim 1: For all $A \in N$ and all $i \in I$, if $A^* \xRightarrow{*} H \in \mathcal{H}_T$ is a derivation in HRG and $PROP(H, i)$ holds, then there is a derivation $A_i^* \xRightarrow{*} H$ in HRG' .

Proof of Claim 1 (by induction on the number of steps in the derivation):

Let $A^* \xRightarrow{1} H \in \mathcal{H}_T$ be a one-step derivation in HRG and $PROP(H, i)$ be satisfied. Then we can assume that $(A, H) \in P$ and $H = REPLACE(H, repl)$ with $repl(e) = l_H(e)^*$ for $e \in E_H$. By $\mathcal{C}, PROP'$ -compatibility of $PROP$, there is a mapping $assign : E_H \rightarrow I$ such that $PROP'(H, assign, i)$ and $PROP(l_H(e)^*, assign(e))$ hold for $e \in E_H$. Hence, $(A_i, H_{assign}) \in P'$. Moreover, $H_{assign} = H$. Thus, there is a derivation $A_i^* \xRightarrow{1} H$ in HRG' .

Suppose now that the statement is true for k -step derivations. Let $A^* \xRightarrow{k+1} H \in \mathcal{H}_T$ be a $(k+1)$ -step derivation in HRG and $PROP(H, i)$ be satisfied. Then we can assume that the derivation is of the form $A^* \xRightarrow{k} R \xRightarrow{1} H$ for some $(A, R) \in P$. Consider now the fibres $l_R(e)^* \xRightarrow{k} H(e)$ of $e \in E_R$. Since $PROP(H, i)$ holds and $PROP$ is $\mathcal{C}, PROP'$ -compatible, there is a mapping $assign : E_R \rightarrow I$ such that $PROP'(R, assign, i)$ and $PROP(H(e), assign(e))$ hold for $e \in E_R$. By construction of P' , $(A_i, R_{assign}) \in P'$. Moreover, by inductive hypothesis, for $e \in E_R$ with $l_R(e) \in N$, there is a derivation $l_R(e)^*_{assign(e)} \xRightarrow{*} H(e)$ in HRG' . Embedding of these derivations into R_{assign} yields a derivation $R_{assign} \xRightarrow{*} H'$. Since R_{assign} and R differ only in the labels of the nonterminal hyperedges and $H = REPLACE(R, repl)$ with $repl(e) = H(e)$ for $e \in E_R$, $H' = REPLACE(R_{assign}, repl)$ equals H . Thus, there is a derivation $A_i^* \xRightarrow{k} R_{assign} \xRightarrow{*} H$ in HRG' .

Claim 2: For all $A \in N$ and all $i \in I$, if $A_i^* \xRightarrow{*} H \in \mathcal{H}_T$ is a derivation in HRG' , then there is a derivation $A^* \xRightarrow{*} H$ in HRG , and $PROP(H, i)$ holds.

Proof of Claim 2 (by induction on the number of steps in the derivation):

Let $A_i^* \xRightarrow{1} H \in \mathcal{H}_T$ be a one-step derivation in HRG' . Then we can assume that $(A_i, H) \in P'$ and $H = REPLACE(H, repl)$ with $repl(e) = l_H(e)^*$ for $e \in E_H$. By construction of P' , there are a production $(A, H) \in P$ and a mapping $assign : E_H \rightarrow I$ such that $PROP'(H, assign, i)$ and $PROP(l_H(e)^*, assign(e))$ hold for $e \in E_H$. Thus, there is a derivation $A^* \xRightarrow{1} H$ in HRG and $PROP(H, i)$ is satisfied.

Suppose that the statement is true for k -step derivations. Let $A_i^* \xRightarrow{k+1} H \in \mathcal{H}_T$ be a $(k+1)$ -step derivation in HRG' . Then we can assume that the derivation decomposes into $A_i^* \xRightarrow{k} R_{assign} \xRightarrow{1} H$ for some $(A_i, R_{assign}) \in P'$. Hence $PROP'(R, assign, i)$ and, for terminal hyperedges $e \in E_R$, $PROP(l_R(e)^*, assign(e))$ hold. For nonterminal hyperedges $e \in E_{R_{assign}}$, the fibres of the derivation $R_{assign} \xRightarrow{k} H$ are of the form $l_R(e)^*_{assign(e)} \xRightarrow{k} H(e)$.

By inductive hypothesis, for each nonterminal $e \in E_R$, there is a derivation $l_R(e)^* \xRightarrow{*} H(e)$ in HRG and $PROP(H(e), assign(e))$ holds. Embedding the derivations into R yields a derivation $R \xRightarrow{*} H$ in HRG . Thus, there is a derivation $A^* \Rightarrow R \xRightarrow{*} H$ in HRG . Moreover, by $C, PROP'$ -compatibility of $PROP$, $PROP(H, i)$ holds. This completes the inductive step.

By Claim 1 and 2, for $l(Z) \in N$ and $i_0 \in I$, there is a derivation $l(Z)_{i_0}^* \xRightarrow{*} H$ in HRG' if and only if there is a derivation $l(Z)^* \xRightarrow{*} H \in \mathcal{H}_T$ in HRG and $PROP(H, i_0)$ is satisfied. This completes the proof of the theorem. $\square$

Remark

In the proof of Theorem 3.9, we gave an effective construction of a hyperedge-replacement grammar HRG_{PR} which generates all hypergraphs in $L(HRG)$ satisfying the predicate PR . Moreover, it has been shown that the Emptiness Problem for hyperedge-replacement grammars is decidable (see Corollary IV.1.3). Combining these results, the questions

- (1) Is there a hypergraph in the language $L(HRG)$ having the property PR ?
- (2) Do all hypergraphs in the language $L(HRG)$ have the property PR ?

may be handled by investigating the questions

- (1') Is the language $L(HRG_{PR})$ non-empty?
- (2') Is the language $L(HRG_{\neg PR})$ empty?

These ideas are due to Rozenberg and Welzl [RW 86b] and Courcelle [Co 87].

3.10 Corollary (metatheorem for decision problems)

Let PR be $\mathcal{C}$ -compatible with respect to some class $\mathcal{C}$ of hyperedge-replacement grammars. Then, for all $HRG \in \mathcal{C}$, it is decidable whether

- (1) PR holds for all $H \in L(HRG)$ except perhaps a finite number,
- (2) PR holds for no $H \in L(HRG)$ except perhaps a finite number.

Proof

By Corollary IV.2.5, the Finiteness Problem for hyperedge-replacement languages is decidable. Therefore, we may construct the hyperedge-replacement grammars HRG_{PR} and $HRG_{\neg PR}$ and test whether $L(HRG_{\neg PR})$ is resp. $L(HRG_{PR})$ is finite. $\square$

Non-Compatible Graph-Theoretic Properties

Theorem 3.9 can be used to show that there exist properties which are not compatible in the sense of Definition 3.1.

3.11 Theorem (square number of nodes)

The property “ $SQUARE$ ” given by

$SQUARE(H)$ if and only if there is an $n \in \mathbb{N}$ such that $|V_H| = n^2$ for $H \in \mathcal{H}_C$ is not $\mathcal{HRG}$ -compatible.

Proof

Assume that *SQUARE* is *HRG*-compatible. Consider now a hyperedge-replacement grammar *HRG* which generates the set of all discrete graphs:

$$L(HRG) = \{DISCRETE_k \mid k \geq 1\}.$$

By Theorem 3.9, there is a hyperedge-replacement grammar *HRG_{SQUARE}* which generates $L(HRG_{SQUARE}) = \{DISCRETE_k \mid k \text{ is square}\}$. On the other hand, we know that the set of discrete graphs with square number of nodes cannot be generated by a hyperedge-replacement grammar, because the growing with respect to the number of nodes is not sublinear as required by Corollary IV.2.6 (sublinear growing). Thus, *SQUARE* cannot be *HRG*-compatible. $\square$

Unsolvable Graph-Theoretic Problems

There are questions referring to single hypergraphs only which have the same form as compatible properties, but are undecidable. We present the following example of this type:

3.12 Theorem (non-trivial automorphism group)

Given an arbitrary hyperedge-replacement grammar *HRG*, it is undecidable whether there is a hypergraph $H \in L(HRG)$ with a non-trivial automorphism group.

Proof

For each context-free string grammar *CFSG*, there is a hyperedge-replacement grammar obtained as follows: Each production (A, w) is transformed into a production (A, w^*) where w^* is a directed path whose edges are labeled with the letters of w . A further transformation yields the hyperedge-replacement grammar *HRG*: The terminals of *CFSG* are added to the nonterminals, and, for each terminal a of *CFSG*, we have a new terminal symbol a' and a production $(a, R(a))$ where $R(a)$ consists of two parallel edges in opposite directions each labeled with a' . Then there is a $H \in L(HRG)$ with non-trivial automorphism group if and only if $L(CFSG)$ contains a palindrome. But this is undecidable, as a straightforward application of the Post Correspondence Problem shows. $\square$

3.13 Related Investigations

In this section, a spectrum of graph-theoretic properties has been studied and proved to be compatible. Moreover, the set of compatible properties has been shown to be closed under Boolean operations. Using the decidability result for compatible properties saying that if a graph-theoretic property is compatible with the derivation process, it is decidable whether some or all generated graphs and hypergraphs have this property; thus, all considered properties are shown as decidable.

Related investigations can be found in the literature.

(1) Janssens and Rozenberg ([JR 81]) investigate — beside some standard language theoretic decision problems — graph-theoretic decision problems and present two basic techniques for proving the undecidability of some problems concerning node-label-controlled (NLC) graph grammars. The first one consists of a rather elaborate way of coding (an

instance of) the Post Correspondence Problem into the language of a NLC grammar. The second one consists of reducing the actual decision problem to the Emptiness Problem for context-sensitive string grammars. Using these techniques, the questions

"Does the language of a NLC grammar contain a totally disconnected graph?",

"Does it contain a connected graph, a planar graph, a Hamiltonian graph, etc.?"

turn out to be undecidable. Only the bounded-degree problem for NLC grammars "Given a graph language L , is there a bound for the degree of each node of each graph $G \in L$?" can be shown to be decidable (see [JRW 86]).

(2) In [RW 86b], Rozenberg and Welzl restrict to boundary node-label-controlled (BNLC) graph grammars (the generative power of which is of a context-free nature) and study questions of the form

"If L is a BNLC language and PR a graph-theoretic property, is the set of all graphs L_{PR} from L satisfying PR again a BNLC language, too?"

Then the question

"Does the language of a BNLC grammar contain a graph satisfying PR ?"

reduces to the questions "Is L_{PR} a BNLC language?" and "Is $L_{PR} \neq \emptyset$?". With respect to the first question, they give affirmative answers for k -colorability, connectivity, and subcontraction. The second question is decidable for all BNLC languages. Thus, in [RW 86b], Rozenberg and Welzl present a technique for proving the decidability of some graph-theoretic problems concerning BNLC grammars.

(3) In [Co 87b], Courcelle presents a similar metatheorem for decision problems for a certain type of hyperedge-replacement grammars using the quite different framework of algebraic systems. Properties *definable* by second-order formulas are shown to behave like compatible properties with respect to the decision problems considered in [HKV 87] and this thesis. Although Courcelle's results seem to indicate that definable properties are compatible, the relationship between both concepts is not yet revealed fully.

(4) Moreover, in a paper by Lengauer and Wanke [LW 88] decision problems similar to ours are studied. They use a type of graph grammars which are specific cases of hyperedge-replacement grammars and consider so-called *finite* properties which are defined independent of the derivation process. The relationship between finite properties and compatible properties will need a thorough investigation.

(5) The concepts and techniques considered in this section apply to complexity issues. The compatibility says that a property can be tested for a hypergraph derived by long fibres (starting from a handle) by testing some hypergraphs derived by shorter fibres. This observation leads to fast tests of compatible properties under specific conditions. The details are worked out — in some degree — in [Kr 87]. Other authors use more or less the same ideas to come up with efficient solutions of particular graph problems on some types of graphs. In [Sl 82], Slisenko introduces a class of graph grammars, which can be seen as special hyperedge-replacement grammars, and gives a polynomial-time algorithm testing the existence of Hamiltonian paths in the generated graphs. Lengauer (see, e.g., [Le 85]) and Lautemann (see [La 88a]) present efficient tests for a variety of graph problems dealing with hierarchical graphs and decomposition trees respectively.

4. Some Compatible Graph-Theoretic Functions

A hyperedge-replacement grammar, as a generating device, specifies a (hyper)graph language. Unfortunately, in a finite amount of time, the generating process only produces a finite section of the language explicitly (and even this may consume much time). Hence one may wonder what the hyperedge-replacement grammar can tell us about the generated language. As a matter of fact, by Theorem 1.1, we have the following nice situation. Given a hyperedge-replacement grammar and an arbitrary terminal (hyper)graph H with derivation $A^* \Rightarrow R \xRightarrow{*} H$, we get a decomposition of H into “smaller” components which are derivable from the handles of the hyperedges in R . If one is interested in values of graph-theoretic functions of derived (hyper)graphs, one may ask how a certain value of a derived (hyper)graph depends on values of the components. A function is said to be “compatible” with the derivation process of hyperedge-replacement grammars if it can be computed for each derived (hyper)graph H by computing the values (or related values) for the components and composing the values to the value of H .

In this section, we pick up several graph-theoretic functions and show that they are “compatible” with the replacement process of hyperedges. A formal definition of compatibility with respect to functions is given in the next section. We discuss the number of nodes and hyperedges, the number of paths and cycles, the length of a shortest path, the length of a longest simple path, the minimum and maximum degree, and the number of components.

Let $HRG = (N, T, P, Z)$ be a typed and well-formed hyperedge-replacement grammar, $H \in \mathcal{H}_T$, $A^* \Rightarrow R \xRightarrow{*} H$ a derivation of H in HRG , and, for $e \in E_R$, $l_R(e)^* \xRightarrow{*} H(e)$ be the fibre of $R \xRightarrow{*} H$ induced by e . Then the number of nodes in H can be computed from the number of nodes in R and the number of internal nodes in the $H(e)$'s. Similarly, the number of internal nodes can be computed. Even simpler, the number of hyperedges in H can be determined by the number of hyperedges in the $H(e)$'s.

4.1 Theorem (Number of Nodes and Hyperedges)

For a hypergraph $H \in \mathcal{H}_G$, let $|V_H|$ denote the number of nodes, $|INT_H|$ the number of internal nodes, and $|E_H|$ the number of hyperedges in H . Then

$$\begin{aligned} |V_H| &= |V_R| + \sum_{e \in E_R} |INT_{H(e)}|, \\ |INT_H| &= |INT_R| + \sum_{e \in E_R} |INT_{H(e)}|, \\ |E_H| &= \sum_{e \in E_R} |E_{H(e)}|. \end{aligned}$$

Proof

Without loss of generality, we may assume $H = \text{REPLACE}(R, \text{repl})$ with $\text{repl}(e) = H(e)$ for $e \in E_R$ (cf. Theorem 1.1). By definition of replacement, $V_H = V_R + \sum_{e \in E_R} \text{INT}_{H(e)}$, $E_H = \sum_{e \in E_R} E_{H(e)}$, and $\text{EXT}_H = \text{EXT}_R$. Hence, $\text{INT}_H = V_H - \text{EXT}_H = V_R - \text{EXT}_R + \sum_{e \in E_R} \text{INT}_{H(e)}$. Now the cardinality statements with respect to the number of nodes and the number of hyperedges follow directly from the set-theoretic statements. $\square$

Remarks

1. Similarly, the composed function *size* given by $\text{size}(H) = |V_H| + |E_H|$ can be handled. It makes use of the auxiliary function *intsize* given by $\text{intsize}(H) = |\text{INT}_H| + |E_H|$.

$$\begin{aligned}\text{size}(H) &= \text{size}(R) + \sum_{e \in E_R} \text{intsize}(H(e)), \\ \text{intsize}(H) &= \text{intsize}(R) + \sum_{e \in E_R} \text{intsize}(H(e)).\end{aligned}$$

2. The density function *dens* given by $\text{dens}(H) = \frac{|E_H|}{|V_H|}$, if $|V_H| > 0$ (and $\text{dens}(H) = \diamond$ ¹ otherwise), can also be expressed in such a way:

$$\text{dens}(H) = \frac{\sum_{e \in E_R} |E_{H(e)}|}{|V_R| + \sum_{e \in E_R} |\text{INT}_{H(e)}|}$$

The expression for computing *dens*(*H*) makes use of the possibility to compute the number of internal nodes as well as the number of hyperedges of the *H*(*e*)'s. It does not make use of the density of some of the *H*(*e*)'s. $\square$

For simplifying the technicalities, we restrict our following consideration to the class $\mathcal{ERG}$ of edge-replacement grammars in the sense of [HK 83+85]. Let us remind that a typed and well-formed hyperedge-replacement grammar $HRG = (N, T, P, Z)$ is in $\mathcal{ERG}$ if and only if the right-hand sides of the productions as well as the axiom are (1,1)-graphs. Note that, in this case, each $G \in L(HRG)$ is a (1,1)-graph, i.e., a graph with two distinguished nodes *begin*_{*G*} and *end*_{*G*}.

¹ *dens*, *minpath*, *maxpath*, *mincycle*, and *maxcycle* are defined to be functions with values in $\mathbb{N} \cup \{\diamond\}$, the set of all non-negative integers plus a special symbol $\diamond$. We use this special symbol $\diamond$, if the considered function has no sensible integer value. We calculate with $\diamond$ as follows: $\forall i \in I \ \forall n_i \in \mathbb{N} \cup \{\diamond\}$,

- $\sum_{i \in I} n_i = \diamond$ and $\prod_{i \in I} n_i = \diamond$ if and only if $n_j = \diamond$ for some $j \in I$,
- $\min_{i \in I} n_i = \min_{i \in I'} n_i$ and $\max_{i \in I} n_i = \max_{i \in I'} n_i$ for $I' = \{i \in I \mid n_i \neq \diamond\}$, and $\min_{i \in I} n_i = \diamond$ and $\max_{i \in I} n_i = \diamond$ for $I = \emptyset$.

Let G be a graph. A path joining v_0 and v_n is a sequence $p = v_0, e_1, v_1, e_2, \dots, e_n, v_n$ of alternating nodes and edges such that for $1 \leq i \leq n$, v_{i-1} and v_i are the nodes incident with e_i . If $v_0 = v_n$, then p is said to be a cycle. If in a path each node appears once, then the sequence is called a *simple* path. If each node appears once except that $v_0 = v_n$ and $n \geq 3$, then p is a *simple cycle*. The *length* of a path or a cycle p , denoted $\text{length}(p)$, is the number of edges it contains. " e on p " denotes the fact that e occurs in p .

4.2 Theorem (Number of Simple Paths, Minimum and Maximum Simple-Path Length)

For a (1,1)-graph G , let PATH_G denote the set of simple paths joining begin_G and end_G and $\text{numpath}(G)$ the number of these paths in G . Moreover, let $\text{minpath}(G)$ and $\text{maxpath}(G)$ denote the minimum resp. maximum simple-path length, if any; otherwise, $\text{minpath}(G) = \text{maxpath}(G) = \diamond$. Then

$$\begin{aligned}\text{numpath}(G) &= \sum_{p \in \text{PATH}_R} \prod_{e \text{ on } p} \text{numpath}(G(e)), \\ \text{minpath}(G) &= \min_{p \in \text{PATH}_R} \sum_{e \text{ on } p} \text{minpath}(G(e)), \\ \text{maxpath}(G) &= \max_{p \in \text{PATH}_R} \sum_{e \text{ on } p} \text{maxpath}(G(e)).\end{aligned}$$

Proof

Let G be a (1,1)-graph and $\hat{p}$ be a simple path joining begin_G and end_G . Without loss of generality, we can assume $G = \text{REPLACE}(R, \text{repl})$ with $\text{repl}(e) = G(e)$ for each $e \in E_R$ (cf. 1.1). Particularly, $V_R \subseteq V_G$. Starting from begin_G and running through $\hat{p}$, nodes of R are visited in a certain sequence $v_0, \dots, v_n$. Cutting $\hat{p}$ in these nodes, one gets a sequence of simple paths $p_1, \dots, p_n$ (where p_i connects v_{i-1} and v_i for $i = 1, \dots, n$). For $i = 1, \dots, n$, there exists $e_i \in E_R$ such that p_i belongs to $G(e_i)$. Otherwise, if p_i would be covered by more components, it would visit at least three nodes of R . By the same reason, e_i connects v_{i-1} and v_i (for $i = 1, \dots, n$). Thus, $p = v_0, e_1, \dots, e_n, v_n$ forms a simple path joining begin_R and end_R and for $i = 1, \dots, n$, p_i is a simple path joining $\text{begin}_{G(e_i)}$ and $\text{end}_{G(e_i)}$. Conversely, if $p = v_0, e_1, \dots, e_n, v_n$ is a simple path joining begin_R and end_R and, for $i = 1, \dots, n$, p_i is a simple path joining $\text{begin}_{G(e_i)}$ and $\text{end}_{G(e_i)}$, then the replacement of the edges $e_1, \dots, e_n$ by the corresponding paths $p_1, \dots, p_n$ yields a simple path joining begin_G and end_G . Therefore,

$$\text{PATH}_G = \bigcup_{p \in \text{PATH}_R} \{ \text{replace}(p, \text{path}) \mid \text{path}(e) \text{ is a simple path in } G(e) \text{ joining } \text{begin}_{G(e)}, \text{end}_{G(e)} \},$$

where $\text{replace}(p, \text{path})$ denotes the path obtained from p by replacing all edges e on p by the corresponding paths $\text{path}(e)$.

The statements on the number of simple paths as well as the minimum and maximum simple path length follow immediately from the set-theoretical statement: Since, for a simple

path $p \in PATH_R$, $replace(p, path) \neq replace(p, path')$ if and only if $path(e) \neq path'(e)$ for some e on p , the number of simple paths induced by p is $\prod_{e \text{ on } p} numpath(G(e))$. Since different paths in R yield different paths in G , the number of simple paths in G is the sum of the numbers of simple paths induced by some simple path in R . Since, for a simple path $p \in PATH_R$, $length(replace(p, path)) = \sum_{e \text{ on } p} length(path(e))$, we obtain

$$\min_{p \in PATH_G} length(p) = \min_{p \in PATH_R} \sum_{e \text{ on } p} \min_{path(e) \in PATH_{\alpha(e)}} length(path(e)).$$

Replacing min by max wherever it occurs, we obtain the statement for $\max_{p \in PATH_G} length(p)$. Using the definition of $minpath$ and $maxpath$, we can derive the claimed minimum- and maximum-statements. $\square$

4.3 Theorem (Number of Simple Cycles, Minimum and Maximum Simple-Cycle Length)

For a (1,1)-graph G , let $CYCLE_G$ denote the set of simple cycles and $numcycle(G)$ the number of these cycles in G . Let $mincycle(G)$ and $maxcycle(G)$ denote the minimum resp. maximum simple-cycle length, if any; otherwise, let $mincycle(G) = maxcycle(G) = \circ$. Then

$$\begin{aligned} numcycle(G) &= \sum_{c \in CYCLE_R} \prod_{e \text{ on } c} numpath(G(e)) + \sum_{e \in E_R} numcycle(G(e)), \\ mincycle(G) &= \min \left\{ \min_{c \in CYCLE_R} \sum_{e \text{ on } c} minpath(G(e)), \min_{e \in E_R} mincycle(G(e)) \right\}, \\ maxcycle(G) &= \max \left\{ \max_{c \in CYCLE_R} \sum_{e \text{ on } c} maxpath(G(e)), \max_{e \in E_R} maxcycle(G(e)) \right\}. \end{aligned}$$

Proof

Let G be a (1,1)-graph and $\hat{c}$ be a simple cycle. Without loss of generality, we can assume $G = REPLACE(R, repl)$, with $repl(e) = G(e)$ for each $e \in E_R$ (cf. 1.1). Particularly, $V_R \subseteq V_G$. Starting somewhere on $\hat{c}$ and running through $\hat{c}$, two cases may occur.

First Case: The cycle $\hat{c}$ belongs to a single component, say $G(e)$.

Second Case: The cycle $\hat{c}$ does not belong to a single component. Then we consider the sequence $v_0, \dots, v_n$ of visited nodes of R . Cutting $\hat{c}$ in these nodes, one gets a sequence of simple paths $p_1, \dots, p_{n+1}$ (where p_i connects v_{i-1} and v_i for $i = 1, \dots, n$ and p_{n+1} connects v_n and v_0). For $i = 1, \dots, n+1$, there exists $e_i \in E_R$ such that p_i belongs to $G(e_i)$. Otherwise, if p_i would be covered by more components, it would visit at least three nodes of R . By the same reason, e_i connects v_{i-1} and v_i (for $i = 1, \dots, n+1$). Thus, $c = v_0, e_1, \dots, e_{n+1}, v_{n+1}$ forms a simple cycle in R and, for $i = 1, \dots, n+1$, p_i is a simple path joining $begin_{G(e_i)}$ and $end_{G(e_i)}$.

Conversely, if $c = v_0, e_1, \dots, e_{n+1}, v_{n+1}$ is a simple cycle in R and, for $i = 1, \dots, n+1$, p_i is a simple path joining $begin_{G(e_i)}$ and $end_{G(e_i)}$, then the replacement of the edges

$e_1, \dots, e_{n+1}$ by the paths $p_1, \dots, p_{n+1}$ yields a simple cycle in G . Moreover, if $e \in E_R$ and c is a simple cycle in $G(e)$, then c is a cycle in G , too. Therefore,

$$\begin{aligned} CYCLE_G = & \bigcup_{c \in CYCLE_R} \{ \text{replace}(c, \text{path}) \mid \text{path}(e) \text{ is a simple path in } G(e) \\ & \text{joining } \text{begin}_{G(e)} \text{ and } \text{end}_{G(e)} \} \\ & \cup \bigcup_{e \in E_R} \{ c \mid c \text{ is a simple cycle in } G(e) \}, \end{aligned}$$

where $\text{replace}(c, \text{path})$ denotes the cycle obtained from c by replacing all edges e on c by the corresponding simple paths $\text{path}(e)$.

The statements on the number of simple cycles as well as the minimum and maximum simple cycle length are immediate consequences of the set-theoretical statement: Since, for a simple cycle c in R , $\text{replace}(c, \text{path}) \neq \text{replace}(c, \text{path}')$ if and only if $\text{path}(e) \neq \text{path}'(e)$ for some e on c , the number of simple cycles induced by c is $\prod_{e \text{ on } c} \text{numpath}(G(e))$. Different simple cycles in R yield different cycles in G . Finally, cycles belonging to different components are different. Therefore, we obtain the claimed cardinality statement from the set-theoretic statement.

Since, for a simple cycle $c \in CYCLE_R$, $\text{length}(\text{replace}(c, \text{path})) = \sum_{e \text{ on } c} \text{length}(\text{path}(e))$, we get

$$\min_{c \in CYCLE_G} \text{length}(c) = \min \left\{ \min_{c \in CYCLE_R} \sum_{e \text{ on } c} \min_{\text{path}(e) \in PATH_{G(e)}} \text{length}(\text{path}(e)), \right. \\ \left. \min_{c \in E_R} \min_{c \in CYCLE_{G(e)}} \text{length}(c) \right\}$$

A corresponding statement for $\max_{c \in CYCLE_G} \text{length}(c)$ can be obtained. Using the definition of mincycle and maxcycle , we can derive the claimed minimum- and maximum-statements. $\square$

4.4 Theorem (Minimum and Maximum Degree)

For a graph G , let $\text{mindegree}(G)$ and $\text{maxdegree}(G)$ denote the minimum resp. maximum degree among the nodes of G . Moreover, let $\text{minintdegree}(G)$ denote the minimum degree among the internal nodes of G and $\text{bdegree}(G)$ and $\text{edegree}(G)$ the degree of begin_G resp. end_G . Then

$$\begin{aligned} \text{mindegree}(G) &= \min \left\{ \min_{v \in V_R} D_G(v), \min_{e \in E_R} \text{minintdegree}(G(e)) \right\} \\ \text{minintdegree}(G) &= \min \left\{ \min_{v \in INT_R} D_G(v), \min_{e \in E_R} \text{minintdegree}(G(e)) \right\} \\ \text{maxdegree}(G) &= \max \left\{ \max_{v \in V_R} D_G(v), \max_{e \in E_R} \text{maxdegree}(G(e)) \right\} \\ \text{bdegree}(G) &= D_G(\text{begin}_G) \text{ and } \text{edegree}(G) = D_G(\text{end}_G) \end{aligned}$$

where, for $v \in V_R$, $D_G(v) = \sum_{e \in s_R^{-1}(v)} \text{bdegree}(G(e)) + \sum_{e \in t_R^{-1}(v)} \text{edegree}(G(e))$.

Proof

Let G be a (1,1)-graph and v be a node in G . Without loss of generality, we can assume $G = REPLACE(R, repl)$ with $repl(e) = G(e)$ for each $e \in E_R$ (cf. 1.1). Particularly, $V_R \subseteq V_G$. Thus, the degree of v in G , $d_G(v)$, can be determined as follows:

$$d_G(v) = \begin{cases} D_G(v) & \text{if } v \in V_R \\ d_{G(e)}(v) & \text{if } v \in INT_{G(e)} \text{ for some } e \in E_R \end{cases}$$

$$\text{where } D_G(v) = \sum_{e \in s_R^{-1}(v)} d_{G(e)}(begin_{G(e)}) + \sum_{e \in t_R^{-1}(v)} d_{G(e)}(end_{G(e)}).$$

Using the notations introduced above, we get

$$D_G(v) = \sum_{e \in s_R^{-1}(begin_G)} bdegree(G(e)) + \sum_{e \in t_R^{-1}(begin_G)} edegree(G(e)).$$

By definition of the minimum degree of G and the minimum degree of G among the internal nodes, we get

$$\begin{aligned} \mindegree(G) &= \min_{v \in V_G} d_G(v) \\ &= \min \left\{ \min_{v \in V_R} D_G(v), \min_{e \in E_R} \min_{v \in INT_{G(e)}} d_{G(e)}(v) \right\} \\ &= \min \left\{ \min_{v \in V_R} D_G(v), \min_{e \in E_R} \minintdegree(G(e)) \right\}, \end{aligned}$$

$$\begin{aligned} \minintdegree(G) &= \min_{v \in INT_G} d_G(v) \\ &= \min \left\{ \min_{v \in INT_R} D_G(v), \min_{e \in E_R} \min_{v \in INT_{G(e)}} d_{G(e)}(v) \right\} \\ &= \min \left\{ \min_{v \in INT_R} D_G(v), \min_{e \in E_R} \minintdegree(G(e)) \right\}. \end{aligned}$$

Analogously, by definition of the maximum degree of G and the maximum degree of G among the internal nodes,

$$\begin{aligned} \maxdegree(G) &= \max_{v \in V_G} d_G(v) \\ &= \max \left\{ \max_{v \in V_R} D_G(v), \max_{e \in E_R} \max_{v \in INT_{G(e)}} d_{G(e)}(v) \right\} \\ &= \max \left\{ \max_{v \in V_R} D_G(v), \max_{e \in E_R} \maxintdegree(G(e)) \right\}. \end{aligned}$$

Since $\maxdegree(G(e)) = \max \{ \maxintdegree(G(e)), d_{G(e)}(begin_{G(e)}), d_{G(e)}(end_{G(e)}) \}$ and $d_{G(e)}(begin_{G(e)}), d_{G(e)}(end_{G(e)}) \leq \max_{v \in V_R} D_G(v)$, the last expression can be simplified to $\max \{ \max_{v \in V_R} D_G(v), \max_{e \in E_R} \maxdegree(G(e)) \}$.

Finally, by definition of $bdegree$ and $edegree$, we get $bdegree(G) = D_G(begin_G)$ and $edegree(G) = D_G(end_G)$. This completes the proof. $\square$

4.5 Theorem (Number of Components)

For a $(1,1)$ -graph G , let $comp(G)$ denote the number of connected components in G and $newcomp(G)$ denote the number of connected components neither containing $begin_G$ nor end_G . Then

$$comp(G) = comp(R') + \sum_{e \in E_R} newcomp(G(e)),$$

$$newcomp(G) = newcomp(R') + \sum_{e \in E_R} newcomp(G(e))$$

where $R' = R - \{e \in E_R \mid numpath(G(e)) = 0\}$.

Remark

$newcomp(G)$ can be expressed directly by

$$newcomp(G) = \begin{cases} comp(G) - 1 & \text{if } numpath(G) > 0 \\ comp(G) - 2 & \text{otherwise} \end{cases}.$$

Proof

Let $COMP$ be a connected component in G . Without loss of generality, we can assume $G = REPLACE(R, repl)$ with $repl(e) = G(e)$ for each $e \in E_R$ (cf. 1.1). In particular, $V_R \subseteq V_G$. Now two cases may occur.

First Case: $V_{COMP} \subseteq INT_{G(e)}$ for some $e \in E_R$. Then $COMP$ is a "new component" created by e , a component neither containing $begin_{G(e)}$ nor $end_{G(e)}$.

Second Case: $V_{COMP} \subseteq \sum_{e \in F} V_{G(e)}$ with $F = \{e \in E_R \mid V_{COMP} \cap EXT_{G(e)} \neq \emptyset\}$. Consider the set $F' = \{e \in E_R \mid begin_{G(e)}, end_{G(e)} \in V_{COMP} \text{ and } numpath(G(e)) > 0\}$. Then all $e \in F'$ are in R' . Moreover, the subgraph $COMP' \subseteq R'$ consisting of the edges of F' together with its sources and targets is connected because $COMP$ is connected. $COMP'$ forms a component of R' . [Suppose there is a node in $v \in V_R - V_{COMP'}$ which is joined with a node $v' \in V_{COMP'}$ by an edge e in R' . Then $numpath(G(e)) > 0$, which means that v is joined with v' by a path in G . Since v' belongs to $COMP$, v belongs to $COMP$. Therefore, $e \in F'$ and $v \in COMP'$, a contradiction.] Consequently, $comp(G) \leq comp(R') + \sum_{e \in E_R} newcomp(G(e))$. Analogously, $newcomp(G) \leq newcomp(R') + \sum_{e \in E_R} newcomp(G(e))$.

Conversely, let $COMP'$ be a component in R' . Then, for each pair of nodes $v, v' \in V_{COMP'}$, v and v' are connected in G (the nodes are connected in R' and, for all $e \in E_{R'}$, $numpath(G(e))$ is greater than 0). Consider now the edges in $e \in E_R - E_{R'}$. Two cases may occur: (a) The source and target node of e are in the same component of R' . Then all nodes of $G(e)$ which are connected with $begin_{G(e)}$ or $end_{G(e)}$ are in the same component as $s_R(e)$ and $t_R(e)$. (b) The source and the target node are in different components of R' . In any case, the nodes of $G(e)$ which are connected with $begin_{G(e)}$ are in the same component as $s_R(e)$ and all nodes of $G(e)$ which are connected with $end_{G(e)}$ are in the same component as $t_R(e)$. Different components in R' yield different components in G . Moreover, for an edge $e \in E_R$, the components in $G(e)$ which neither contain $begin_{G(e)}$ nor $end_{G(e)}$ yield additional components. Thus, $comp(R') + \sum_{e \in E_R} newcomp(G(e)) \leq comp(G)$ and $newcomp(R') + \sum_{e \in E_R} newcomp(G(e)) \leq newcomp(G)$. $\square$

5. A General View of Compatible Functions

In this section, we introduce the notion of compatible functions in a formal way such that all functions considered in the previous section are special cases. Roughly speaking, a function f_0 on hypergraphs is said to be compatible with the derivation process of hyperedge-replacement grammars if, for each hypergraph H and each derivation of it, the value of H , $f_0(H)$, can be computed from the values of some specific subhypergraphs $H(e)$ determined by the fibres of the derivation. As the examples will show, this view is oversimplified for most applications. To compute the value of H , it might be necessary to compute the values of some other related functions for the $H(e)$'s. Therefore, we use families of functions indexed by some finite set I and we need a mapping *assign* which determines the values for the $H(e)$'s with respect to the different value functions.

The notion of compatible functions generalizes the notion of compatible predicates (see Section 3). More interesting, a certain type of compatible functions that are composed of minima, maxima, sums, and products induce compatible predicates of the form: the function value of a graph exceeds a given fixed integer, or the function value does not exceed a fixed integer. Consequently, we get the decidability of the problems (1) and (2) in the introduction of this chapter for these predicates as a corollary.

5.1 Definition (compatible functions)

1. Let $\mathcal{C}$ be a class of hyperedge-replacement grammars, I a finite index set, VAL a set of values, $f : \mathcal{H}_C \times I \rightarrow VAL$ a function, and f' a function defined on triples $(R, assign, i)$ with $R \in \mathcal{H}_C$, $assign : E_R \times I \rightarrow VAL$, and $i \in I$.¹ Then f is called $(\mathcal{C}, f')$ -compatible if, for all $HRG = (N, T, P, Z) \in \mathcal{C}$ and all derivations of the form $A^* \Rightarrow R \xRightarrow{*} H$ with $A \in N$ and $H \in \mathcal{H}_T$, and for all $i \in I$,

$$f(H, i) = f'(R, assign, i)$$

where $assign : E_R \times I \rightarrow VAL$ is given by $assign(e, j) = f(H(e), j)$ for all $e \in E_R$ and all $j \in I$.

2. A function $f_0 : \mathcal{H}_C \rightarrow VAL$ is called $\mathcal{C}$ -compatible if functions f and f' and an index i_0 exist such that $f_0 = f(-, i_0)$ and f is $(\mathcal{C}, f')$ -compatible.²

Remark

Intuitively, a function is compatible if it can be computed for a large hypergraph derived by a fibre by computing some values for the smaller components of the corresponding shorter fibres. Such a function must be closed under isomorphisms, because the derivability of hypergraphs is independent of the representation of nodes and hyperedges.

¹ We assume that all considered functions are *closed under isomorphisms*, i.e., for a function f , if $H \cong H'$ for some $H, H' \in \mathcal{H}_C$, then $f(H, i) = f(H', i)$ (resp. $f(H, assign, i) = f(H', assign, i)$) for all $i \in I$.

² For $i \in I$, $f(-, i)$ denotes the unary function defined by $f(-, i)(H) = f(H, i)$ for all $H \in \mathcal{H}_C$.

5.2 Examples (compatible functions)

By Theorem 4.1, the following functions on hypergraphs are $\mathcal{HRG}$ -compatible:

- (1) the number of nodes,
- (2) the number of hyperedges,
- (3) the size, and
- (4) the density of a hypergraph.

By Theorems 4.2-4.5, the following functions on graphs are $\mathcal{ERG}$ -compatible:

- (5) the number of simple paths connecting the external nodes,
- (6) the minimum-path length (of paths connecting the external nodes),
- (7) the maximum-simple-path length (of paths connecting the external nodes),
- (8) the number of simple cycles,
- (9) the minimum-cycle length,
- (10) the maximum-simple-cycle length,
- (11) the minimum degree,
- (12) the maximum degree, and
- (13) the number of components of a graph.

We relate now the notion of compatible predicates with compatible functions.

5.3 Theorem (compatible predicates — compatible functions)

Let $PROP_0$ be a $\mathcal{C}$ -compatible predicate. Then the function $f_0 : \mathcal{H}_C \rightarrow \{0, 1\}$ given by

$$f_0(H) = \begin{cases} 1 & \text{if } PROP_0(H) \text{ holds} \\ 0 & \text{otherwise} \end{cases}$$

is $\mathcal{C}$ -compatible.

Proof

Let $PROP_0$ be a $\mathcal{C}$ -compatible predicate, $PROP$, $PROP'$ the corresponding predicates, I the corresponding index set, and i_0 the index such that $PROP_0 = PROP(-, i_0)$. Then we define functions f , f' as follows.

$$f(H, i) = \begin{cases} 1 & \text{if } PROP(H, i) \text{ holds} \\ 0 & \text{otherwise} \end{cases}$$

$$f'(R, assign', i) = \begin{cases} 1 & \text{if } PROP'(R, assign, i) \text{ holds for some } assign : E_R \rightarrow I \\ & \text{with } assign'(e, assign(e)) = 1 \text{ for all } e \in E_R \\ 0 & \text{otherwise} \end{cases}$$

Obviously, $f_0(H) = f(H, i_0)$. Hence, it remains to show that f is $(\mathcal{C}, f')$ -compatible. Let $H \in \mathcal{H}_T$, $A^* \Rightarrow R \xRightarrow{*} H$ a derivation of H in HRG , and, for $e \in E_R$, $l_R(e)^* \xRightarrow{*} H(e)$ be the fibre of $R \xRightarrow{*} H$ induced by e .

If $f(H, i) = 1$, then $PROP(H, i)$ is satisfied. By $(\mathcal{C}, PROP')$ -compatibility of $PROP$, there is a mapping $assign : E_R \rightarrow I$ such that $PROP'(R, assign, i)$ and $PROP(H(e), assign(e))$ for $e \in E_R$ hold. Thus, $f(H(e), assign(e)) = 1$ for $e \in E_R$. Define now $assign' : E_R \times I \rightarrow \{0, 1\}$ by $assign'(e, j) = f(H(e), j)$ for $e \in E_R$ and $j \in I$.

Then $\text{assign}'(e, \text{assign}(e)) = f(H(e), \text{assign}(e)) = 1$ for all $e \in E_R$. Using the definition of f' , we get $f'(R, \text{assign}', i) = 1$ where $\text{assign}'(e, j) = f(H(e), j)$ for $e \in E_R$ and $j \in I$. Conversely, if $f'(R, \text{assign}', i) = 1$, $\text{assign}'(e, j) = f(H(e), j)$ for $e \in E_R$, and $j \in I$, then there is a mapping $\text{assign} : E_R \rightarrow I$ with $\text{assign}'(e, \text{assign}(e)) = 1$ for $e \in E_R$ such that $\text{PROP}'(R, \text{assign}, i)$ holds. By the assumptions, $f(H(e), \text{assign}(e)) = \text{assign}'(e, \text{assign}(e)) = 1$ for $e \in E_R$; thus, $\text{PROP}(H(e), \text{assign}(e))$ holds for all $e \in E_R$. Since PROP is $(\mathcal{C}, \text{PROP}')$ -compatible, $\text{PROP}(H, i)$ holds, i.e., $f(H, i) = 1$. Now $f(H, i) = f'(R, \text{assign}', i)$ where $\text{assign}'(e, j) = f(H(e), j)$ for all $e \in E_R$ and $j \in I$, i.e., f is $(\mathcal{C}, f')$ -compatible. $\square$

Specific $\mathcal{C}$ -compatible functions with values in $\mathbb{N}^\diamond = \mathbb{N} \cup \{\diamond\}$ induce specific $\mathcal{C}$ -compatible predicates.

5.4 Definition (special compatible functions)

1. A function $f : \mathcal{H}_\mathcal{C} \times I \rightarrow \mathbb{N}^\diamond$ is said to be $(\mathcal{C}, \min, \max, +, \cdot)$ -compatible if there exists an f' such that for each right-hand side R of some production in $\mathcal{C}$ and each $i \in I$, $f'(R, -, i)$ corresponds to an expression formed with variables $\text{assign}(e, j)$ ($e \in E_R, j \in I$) and constants from $\mathbb{N}$ by addition, multiplication, minimum, and maximum and f is $(\mathcal{C}, f')$ -compatible. The function is $(\mathcal{C}, \max, +, \cdot)$ -compatible if the operation $\min$ does not occur.
2. A function $f_0 : \mathcal{H}_\mathcal{C} \rightarrow \mathbb{N}^\diamond$ is $(\mathcal{C}, \min, \max, +, \cdot)$ -compatible [resp. $(\mathcal{C}, \max, +, \cdot)$ -compatible] if a function f and an index i_0 exist such that $f_0 = f(-, i_0)$ and f is $(\mathcal{C}, \min, \max, +, \cdot)$ -compatible [resp. $(\mathcal{C}, \max, +, \cdot)$ -compatible].

5.5 Theorem (compatible functions — compatible predicates)

Let $f_0 : \mathcal{H}_\mathcal{C} \rightarrow \mathbb{N}^\diamond$ be a $(\mathcal{C}, \min, \max, +, \cdot)$ -compatible function for some class $\mathcal{C}$ of hyperedge-replacement grammars. Moreover, let $n \in \mathbb{N}^\diamond$. Then the predicates given by “ $f_0(H) \leq n$ ” and “ $f_0(H) > n$ ” are $\mathcal{C}$ -compatible.³

Proof

Let $f_0 : \mathcal{H}_\mathcal{C} \rightarrow \mathbb{N}^\diamond$ be a $(\mathcal{C}, \min, \max, +, \cdot)$ -compatible function, f and f' the corresponding functions, I the corresponding index set and i_0 the index such that $f_0 = f(-, i_0)$. Then we define a new index set $\hat{I}$ and predicates PROP , PROP' as follows.

Let $J = \{0, 1, \dots, n\} \cup \{\diamond, \text{big}\}$ for $n \in \mathbb{N}$ and $J = \{\diamond, \text{big}\}$ for $n = \diamond$. Moreover, let $[-] : \mathbb{N}^\diamond \rightarrow J$ be the mapping with $[m] = \text{big}$ for $m > n$ and $[m] = m$ otherwise. Let $\hat{I}$ be the set of all mappings from I to J , denoted by $\hat{I} = [I \rightarrow J]$.

$\text{PROP}(H, p)$ holds if and only if $[f(H, i)] = p(i)$ for all $i \in I$

$\text{PROP}'(R, \text{assign}, p)$ holds if and only if $[f'(R, \text{assign}', i)] = p(i)$ for all $i \in I$

for all $\text{assign}' : E_R \times I \rightarrow \mathbb{N}$ with $[\text{assign}'(e, j)] = \text{assign}(e)(j)$ for all $e \in E_R, j \in I$.

First, we will show that PROP is $(\mathcal{C}, \text{PROP}')$ -compatible.

If $\text{PROP}(H, p)$ holds, then $[f(H, i)] = p(i)$ for all $i \in I$. By the $(\mathcal{C}, f')$ -compatibility of f , we have $[f'(R, \text{assign}', i)] = [f(H, i)] = p(i)$ for all $e \in E_R$ where $\text{assign}'(e, j) = f(H(e), j)$

³ We assume that, for all $n \in \mathbb{N}^\diamond$, $\diamond \leq n$.

for $e \in E_R$, $j \in I$. Now choose $\text{assign} : E_R \rightarrow [I \rightarrow J]$ as $\text{assign}(e)(j) = [\text{assign}'(e, j)]$. Since f' is formed by addition, multiplication, minimum, and maximum, for all assign' with $[\text{assign}'(e, j)] = \text{assign}(e)(j)$ ($e \in E_R$, $j \in I$), $[f'(R, \text{assign}', i)] = p(i)$ for $i \in I$. Thus, by definition of $PROP'$, $PROP'(R, \text{assign}, p)$ is satisfied.

Conversely, let $\text{assign} : E_R \rightarrow [I \rightarrow J]$ be a mapping such that $PROP'(R, \text{assign}, p)$ and $PROP(H(e), \text{assign}(e))$ for $e \in E_R$ hold. By assumption, $f(H, i) = f'(R, \text{assign}', i)$ with $\text{assign}'(e, j) = f(H(e), j)$ for $e \in E_R$ and $j \in I$. Since $PROP(H(e), \text{assign}(e))$ holds, we have $[f(H(e), j)] = \text{assign}(e)(j)$ for all $j \in I$. Consider now assign' with $\text{assign}'(e, j) = f(H(e), j)$. Then $[\text{assign}'(e, j)] = [f(H(e), j)] = \text{assign}(e)(j)$. Using the definition of $PROP'$, we obtain $f'(R, \text{assign}', i) = p(i)$ for all $i \in I$. Thus, $f(H, i) = f'(R, \text{assign}', i) = p(i)$ for all $i \in I$. By Definition of $PROP$, $PROP(H, p)$ is satisfied.

Thus, $PROP$ is $(\mathcal{C}, PROP')$ -compatible. In particular, for each $p : I \rightarrow J$, $PROP(-, p)$ is $\mathcal{C}$ -compatible. Now,

$$\begin{aligned} f_0(H) \leq n &\iff \bigvee_{p: I \rightarrow J \text{ with } p(i_0) \leq n} PROP(H, p), \\ f_0(H) > n &\iff \bigvee_{p: I \rightarrow J \text{ with } p(i_0) = \text{big}} PROP(H, p). \end{aligned}$$

Since the predicates $PROP(-, p)$ are $\mathcal{C}$ -compatible and since $\mathcal{C}$ -compatible predicates are closed under disjunctions (see 3.3), the predicates given by “ $f_0(H) \leq n$ ” and “ $f_0(H) > n$ ” are $\mathcal{C}$ -compatible. $\square$

5.6 Corollary (metatheorem for decision problems)

Let f_0 be a $(\mathcal{C}, \min, \max, +, \cdot)$ -compatible function for some class $\mathcal{C}$ of hyperedge-replacement grammars. Moreover, let $n \in \mathbb{N}^\circ$. Then, for all $HRG \in \mathcal{C}$, the following statements hold:

- (1) It is decidable whether there is some $H \in L(HRG)$ with $f_0(H) \leq n$.
- (2) It is decidable whether, for all $H \in L(HRG)$, $f_0(H) \leq n$.
- (3) It is decidable in linear time whether a generated hypergraph $H \in L(HRG)$ represented by a derivation (resp. a derivation tree) has a value $f_0(H) \leq n$.

Proof

Corollary 5.6 follows immediately from the $\mathcal{C}$ -compatibility of the predicate “ $f_0(-) \leq n$ ” (see Theorem 5.5) and the theorems for $\mathcal{C}$ -compatible predicates given in Section 3. $\square$

A Metatheorem for Boundedness Problems

Given a graph-theoretic function f_0 and a class $\mathcal{C}$ of hyperedge-replacement grammars, we are going to study the following type of questions for all $HRG \in \mathcal{C}$:

“Is it decidable whether the values of all hypergraphs generated by HRG are bounded?”

The question turns out to be decidable provided that f_0 is $(\mathcal{C}, \max, +, \cdot)$ -compatible. We call this result “metatheorem” because of its generic character: Whenever one can prove the $(\mathcal{C}, \max, +, \cdot)$ -compatibility of a function (and we have given various examples in Section 4), one gets a particular decision result for this function as corollary of the metatheorem.

5.7 Theorem (metatheorem for boundedness problems)

Let f_0 be a $(\mathcal{C}, \max, +, \cdot)$ -compatible function for some class $\mathcal{C}$ of hyperedge-replacement grammars. Then, for all $HRG \in \mathcal{C}$, it is decidable whether or not there is a natural number $n \in \mathbb{N}$ such that $f_0(H) \leq n$ for all $H \in L(HRG)$.

Remark

The proof of Theorem 5.7 can be found in [HKV 89].

Combining the compatibility results of Section 4 and Theorem 5.7, one obtains a list of decidability results concerning boundedness problems.

5.8 Corollary (decidability of special boundedness problems)

For each edge-replacement grammar $ERG \in \mathcal{ER}\mathcal{G}$ and each function in the following list, it is decidable whether (or not) the function values of the graphs in $L(ERG)$ grow beyond any bound.

- (1) the number of nodes,
- (2) the number of edges,
- (3) the size,
- (4) the number of simple paths connecting the external nodes,
- (5) the number of simple cycles,
- (6) the maximum-simple-path length (of paths connecting the external nodes),
- (7) the maximum-simple-cycle length, and
- (8) the maximum degree of a graph.

Proof

The statements (1)-(8) follow directly from the theorems 4.1-4.4 and 5.7. □

Remarks

1. Remember that the functions “number of nodes”, “number of hyperedges”, and “size” are compatible for arbitrary hyperedge-replacement grammars $HRG \in \mathcal{HR}\mathcal{G}$.
2. Although we avoided the troublesome technicalities in this paper, we are convinced that the other considerations of this section work for more general types of hyperedge-replacement grammars, too. For example, all the statements should hold even if the class $\mathcal{ER}\mathcal{G}$ is replaced by the class of all hyperedge-replacement grammars which generate ordinary graph languages and use hyperedges with a bounded number of tentacles as nonterminals. We even think that the considered functions are compatible for arbitrary hyperedge-replacement grammars if their definition is properly adapted to hypergraphs.

Finally, let us mention that some problems — like the connectivity problem, the maximum-clique-size problem, and the chromatic-number problem — are trivial in the following sense: for all hyperedge-replacement grammars HRG , there is a bound (depending only on HRG) such that the function values of all graphs do not exceed this bound (see Chapter IV, Section 4). This can be used to show that other boundedness problems — as the minimum-clique-covering problem and the maximum-independent-set problem — are decidable.

The *clique partition number* of a graph G , $C(G)$, is the smallest number of cliques that form a partition of the node set V_G . A set of nodes in a graph G is *independent* if no two of them are adjacent. The largest number of nodes in such a set is called the *independence number* of G and is denoted by $I(G)$.

5.9 Theorem (decidability of special boundedness problems)

For each hyperedge-replacement grammar $HRG \in \mathcal{HRG}$ generating a set of graphs, it is decidable whether

- (1) the clique partition number,
 - (2) the independence number
- of graphs in $L(HRG)$ grows beyond any bound.

Proof

(1) Since for each hyperedge-replacement grammar HRG , the maximum clique size is bounded on $L(HRG)$, say by $c(HRG) \geq 1$, and, for each $G \in L(HRG)$,

$$\frac{|V_G|}{c(HRG)} \leq C(G) \leq |V_G|,$$

the clique partition number is bounded on $L(HRG)$ if and only if the number of nodes is bounded on $L(HRG)$.

(2) Since for each hyperedge-replacement grammar HRG the chromatic number is bounded on $L(HRG)$, say by $k(HRG) \geq 1$, for each $H \in L(HRG)$, the maximum number of equally colored nodes in a $k(HRG)$ -coloring of G , $MAX(G)$, is a lower bound of $I(G)$. On the other side, $|V_G| \leq k(HRG) \cdot MAX(G)$. Thus,

$$\frac{|V_G|}{k(HRG)} \leq I(G) \leq |V_G|.$$

Therefore, the independence number is bounded on $L(HRG)$ if and only if the number of nodes is bounded on $L(HRG)$. □

5.10 Discussion

Each class $\mathcal{C}$ of graph grammars and each function f on graphs with integer values establish a Boundedness Problem:

Is it decidable, for all graph languages $L(GG)$ generated by GG in $\mathcal{C}$, whether there is a bound n such that $f(G) \leq n$ for all $G \in L(GG)$?

In this section, we have been able to show that the Boundedness Problem is solvable for classes of hyperedge-replacement grammars and functions that are compatible with the derivation process and where the values of derivable graphs are composed of maxima, sums, and products of component values. Although this result applies to a variety of examples, it seems to be strangely restricted. Further research should clarify this situation:

- (1) We would expect that the metatheorem holds under more general or modified assumptions. In particular, we would like to know how functions given by minima, differences, or divisions work.

- (2) We suppose that certain combinations of arithmetic operations are not allowed. For instance, maxima and minima seem to antagonize each other — at least sometimes.
- (3) Compatible functions are defined for arbitrary domains. But we have got significant results only for boolean and integer values. What about other domains? How can be arbitrary compatibility be exploited? How do other meaningful interpretations look like?

6. Bibliographic Note

The results presented in Section 2 were published in [Ha 89]. Theorem 3.3 can be found in [Ha 89], Theorem 3.7 corresponds to Theorems 4.3 and 4.4 in [HKV 87], and the theorems of Sections 4 and 5 can be found [HKV 89].

VI. Extensions and Variations of HRG's

In this chapter, extensions and variations of hyperedge-replacement grammars are considered. On the one hand, one may change over to arbitrary hypergraph-replacement grammars in which hypergraphs (not only hyperedges) are replaced. On the other hand, one may equip hyperedge-replacement grammars (as well as hypergraph-replacement grammars) with application conditions or may pass over to parallel hyperedge-replacement grammars. Beside these extensions, one may modify hyperedge-replacement grammars in such a way that the generated objects are no longer hypergraphs but 2-dimensional line figures.

In Section 1, hypergraph-replacement grammars in the sense of the "Berlin approach" to graph grammars (see, e.g., [EPS 73], [Ro 75], [Eh 79]) are considered. As in formal language theory, one may define "monotone", "context-sensitive", and "context-free" grammars. Using the same techniques as in the string case for Chomsky grammars one can show that the membership problem is not decidable for arbitrary hypergraph grammars (corresponding to Type 0 grammars) but it is decidable for monotone hypergraph grammars. Somewhat surprising, monotone hypergraph grammars are more powerful than context-sensitive ones. Context-free hypergraph grammars turn out to be equivalent to hyperedge-replacement grammars. In this sense, hypergraph-replacement grammars generalize hyperedge-replacement grammars. In Section 2, hypergraph-replacement grammars are extended by a very general notion of application conditions which can be defined separately for each production. In addition to the general concept, we study some special cases of hypergraph grammars with application conditions with respect to their generative power. In Section 3, the parallel hyperedge-replacement grammars — closely related to ETOL systems — are discussed. Finally, in Section 4, hyperedge-replacement grammars are equipped with layout information. Therefore, the generated graphs and hypergraphs can easily be displayed as 2-dimensional (or likewise as 3-dimensional) patterns on a "screen", which is given by a layout of the initial hypergraph. In this way, the well studied graph-grammar approach of hyperedge-replacement grammars can be utilized for producing pretty patterns.

1. Hypergraph–Replacement Grammars

This section generalizes hyperedge–replacement grammars to arbitrary hypergraph–replacement grammars which can be seen as a hypergraph version of the “Berlin” graph–grammar approach (see, e.g., Ehrig [Eh 79]).

Informally, the approach works as follows: A hypergraph production $p = (L \supseteq K \subseteq R)$ consists of two hypergraphs L and R and a corresponding interface hypergraph K (also called *gluing hypergraph*) which relates some nodes and hyperedges of L to nodes and hyperedges in R . To apply a production $p = (L \supseteq K \subseteq R)$ to a hypergraph H we have to specify L as a subhypergraph of H and to make sure that the following *contact condition* is satisfied: Each contact point is a gluing point, where a contact point is a node in L which is source or target of a tentacle of a hyperedge in H not belonging to L . The application of a production $p = (L \supseteq K \subseteq R)$ to a hypergraph H leads to a derived hypergraph H' which can be obtained from H with specific subhypergraph L satisfying the contact condition in the following way:

Remove all nodes and hyperedges of L which are not gluing items in H to obtain the persistent hypergraph D (which becomes a hypergraph by the contact condition).

Add all nodes and hyperedges of R which are not gluing items — the gluing items are already in L and hence in D — such that R becomes a subhypergraph of the resulting hypergraph H' .

1.1 Definition (productions and derivations)

1. A *production* $p = (L \supseteq K \subseteq R)$ consists of two hypergraphs L and R , called the *left-hand side* and the *right-hand side* of p , respectively, and an auxiliary hypergraph K , called *gluing hypergraph*, satisfying $K \subseteq L$ and $K \subseteq R$.

2. Let H, H' be hypergraphs and $p = (L \supseteq K \subseteq R)$ be a production. Then H *directly derives* H' *through* p , if H' is isomorphic to the hypergraph X constructed in the following four steps:

(1) CHOOSE a morphism $h : L \rightarrow H$, called the (*left*)–*occurrence map*.

(2) CHECK the *contact condition* and the *identification condition*:

If the image of a node $v \in V_L$ under h contacts some hyperedge $e \in E_H - E_{h(L)}$, i.e., if a tentacle of e grips to it, then $v \in V_R$ (and V_R).

If different items $x, y \in L$ ¹ are identified by h , i.e., if $x \neq y$ but $h(x) = h(y)$, then $x, y \in K$.

(3a) REMOVE the occurrence $h(L)$ of L in H up to $h(K)$ from H yielding the *remainder* $D = H - [h(L) - h(K)]$.²

(3b) ADD the right-hand side R up to K to D yielding the *gluing* $X = D + [R - K]$ ³ the components of which are defined as follows:

¹ “ $x \in L$ ” is a short denotation for “ $x \in V_L$ or $x \in E_L$ ” and is used whenever nodes and hyperedges do not have to be distinguished. Similarly, “ $h(x)$ ” is short for “ $h_V(x)$ or $h_E(x)$ ”.

² “ $-$ ” denotes the difference of sets for nodes and hyperedges separately.

³ “ $+$ ” denotes the disjoint union of sets for nodes and hyperedges separately.

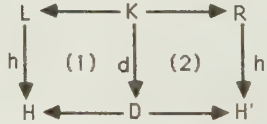
- $V_X = V_D + [V_R - V_K]$ and $E_X = E_D + [E_R - E_K]$,
- each node and each hyperedge keeps its label,
- each hyperedge of E_D keeps its sources and targets,
- each hyperedge of $E_R - E_K$ keeps its sources and targets provided that they are in $V_R - V_K$; otherwise sources and targets are handed over to V_D : $s_X(e) = h'_V(s_R(e))$ and $t_X(e) = h'_V(t_R(e))$ for $e \in E_R - E_K$ where $h'_V : V_R \rightarrow V_X$ is given by $h'_V(v) = v$ for $v \in V_R - V_K$ and $h'_V(v) = h_V(v)$ for $v \in V_K$.

Such a *direct derivation* is denoted by $H \Rightarrow H'$ through p (based on h) or $H \xRightarrow[p]{p} H'$.

3. A sequence of direct derivations of the form $H_0 \xRightarrow[p_1]{p_1} H_1 \xRightarrow[p_2]{p_2} \dots \xRightarrow[p_k]{p_k} H_k$ constitutes a *derivation from H_0 to H_k through $p_1, \dots, p_k$* . Such a derivation may be denoted by $H_0 \xRightarrow[p]{p_1, \dots, p_k} H_k$ if $p_1, \dots, p_k \in P$.

Remarks

1. The occurrence map $h : L \rightarrow H$ locates the occurrence $h(L)$ of L in H .
2. The contact condition ensures that D is a subhypergraph of H .
3. Let $D = G - [h(L) - h(K)]$. Then, for $x \in K$, $d(x) = h(x)$ defines a morphism $d : K \rightarrow D$ locating $h(K)$ in D .
4. Let H' be isomorphic to X and $i : X \rightarrow H'$ be the corresponding isomorphism. Then h' given by $h'(x) = i(x)$ if $x \in R - K$ and $h'(x) = i(d(x))$ otherwise defines a morphism $h' : R \rightarrow H'$, called the *right-occurrence map*. Respectively, $h'(R)$ is called the *occurrence of R in H'* . Further the restriction of $i : X \rightarrow H'$ to $D \subseteq X$ defines a morphism.
5. All morphisms involved in a direct derivation can be grouped into two squares,



which are actually gluing diagrams in the sense of [Eh 79], 2.6. The explicit construction of the derived hypergraph given above corresponds to the gluing analysis and the gluing construction in [Eh 79], 3.7 and 3.8.

6. The definition of a direct derivation is symmetric in the following sense. Let $H \Rightarrow H'$ be a direct derivation through $p = (L \supseteq K \subseteq R)$ based on the (left-)occurrence map $h : L \rightarrow H$. Let $h' : R \rightarrow H'$ be the corresponding right-occurrence map. Then there is a direct derivation $H' \Rightarrow H$ through the *inverse production* $p^{-1} = (R \supseteq K \subseteq L)$ based on the occurrence map $h' : R \rightarrow H'$.

1.2 Definition (hypergraph grammars and languages)

1. A *hypergraph grammar* is a system $HG = (N, T, P, Z)$ where $N \subseteq C$ is a set of nonterminals, $T \subseteq C$ is a set of terminals, P is a finite set of hypergraph productions, and Z is a hypergraph, called axiom.
2. The *hypergraph language $L(HG)$ generated by HG* consists of all terminal labeled hypergraphs which can be derived from the axiom Z by applying productions of P .

1.3 Examples (generation of special sets of graphs)

1. The set of all (directed, edge-labeled) **graphs** over a finite alphabet T can be generated from the graph with empty node set, $\emptyset$, using a production which allows to generate nodes and a production which allows to insert an edge between two given nodes.

$$\begin{aligned}
 p_1 &= \left(\emptyset \supseteq \emptyset \subseteq \bullet \right) \\
 p_2 &= \left(\begin{matrix} \bullet & \bullet \\ 1 & 2 \end{matrix} \supseteq \begin{matrix} \bullet & \bullet \\ 1 & 2 \end{matrix} \subseteq \begin{matrix} \bullet & \bullet \\ 1 & 2 \end{matrix} \xrightarrow{a} \bullet \bullet \right) \quad (\text{for } a \text{ in } T)
 \end{aligned}$$

Note that the second production is neither context-free nor context-sensitive in the sense of Definition 1.4.

2. The set of all (undirected, unlabeled) **connected graphs** with at least one node can be generated from a graph with one node using (1) a production which allows to create a new node and to join it with a node of a connected graph using an edge and (2) a production which allows to join two nodes of a connected graph by an edge.

For generating the set of all (undirected, unlabeled) **Eulerian graphs** (without loops), we recall the fact that a graph is Eulerian if and only if it is connected and every node has even degree. We make use of this relationship and equip each node with a loop the label of which is **even** if the degree of the node is even and **odd** if the degree is odd. Obviously, a new node is connected to the remainder by one edge; therefore the degree is odd. Inserting a new edge between two distinct nodes changes the degree of the involved nodes: if the degree is even, it becomes odd; if it is odd, it becomes even. Removing the degree-information “even” (resp. the loop with label **even**) from a node, the degree of the node cannot be changed any more. It turns out that in a graph without **even**- or **odd**-labeled edges, all nodes have even degree. Therefore, the set of all Eulerian graph can be generated by the grammar $GG = (\{\text{even}, \text{odd}\}, T, P, Z)$ with the axiom

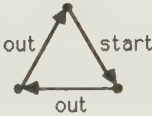


and the productions

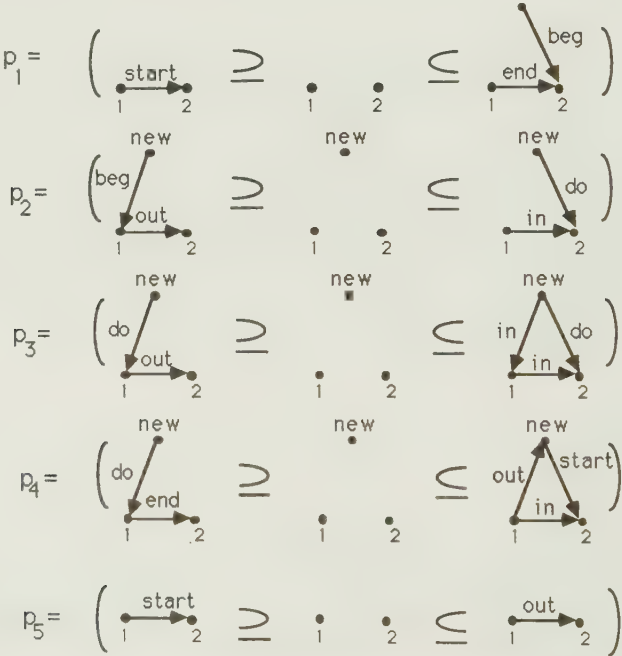
$$\left. \begin{aligned}
 p_1 &= \left(\begin{matrix} \bullet \\ \text{even} \end{matrix} \supseteq \bullet \subseteq \begin{matrix} \bullet & \bullet \\ \text{even} & \text{odd} \end{matrix} \right) \\
 p_2 &= \left(\begin{matrix} \bullet & \bullet \\ \text{even} & \text{even} \end{matrix} \supseteq \begin{matrix} \bullet & \bullet \\ 1 & 2 \end{matrix} \subseteq \begin{matrix} \bullet & \bullet \\ \text{even} & \text{even} \end{matrix} \right) \\
 p_3 &= \left(\begin{matrix} \bullet \\ \text{even} \end{matrix} \supseteq \bullet \subseteq \bullet \right)
 \end{aligned} \right\} \quad \begin{aligned} &\text{where } x, y \in \{\text{even}, \text{odd}\}, \\ &\overline{\text{even}} = \text{odd}, \text{ and } \overline{\text{odd}} = \text{even} \end{aligned}$$

3. The set of all **complete graphs** (with at least three nodes) can be generated from a complete graph with three nodes using the following idea. Given a graph with directed

Hamiltonian cycle. Assume that the edges on the cycle are labeled by *out* (except one distinguished edge with label *beg*) and all other edges are labeled by *in*. Then we may proceed as follows: We start with the *beg*-edge, create a new node, connect it with the target of the *beg*-edge, and label it by *start*. We run through the cycle and connect all nodes on the cycle with the new node. At the end, we create an edge with label *out* and one with label *beg*. The *beg*-edge allows to continue in this way or to finish. The grammar with the axiom



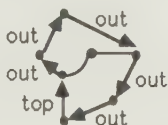
and the productions



generates the set of complete graphs in which a Hamiltonian cycle is indicated by the edge labels *out* and all edges not on the cycle are labeled by *in*. If we add productions which allow to replace *out*-edges as well as *in*-edges by usual (unlabeled) ones, we obtain a grammar which generates the set of all (unlabeled) complete graphs. If a production is added which allows to delete *in*-edges, we obtain a grammar which generates the set of all *Hamiltonian graphs*.

4. Using similar ideas, one can show that the set of all *square grids* can be generated by a (hyper)graph grammar.

5. The set of *snails* (see Chapter IV) can be generated marking the “outer” edges by *out* and the “topical” one by *top*. If an outer edge follows the topical one, we create two new nodes and three new edges, two labeled by *out*, one by *top*; the old *top*-edge and the old *out*-edge become usual edges. To finish the derivation process, the topical edge gets the label *out*. As axiom we choose the snail with one cell.



The productions look as follows.

$$\begin{aligned}
 p_1 &= \left(\begin{array}{c} 1 \text{---} \text{top} \text{---} 2 \\ \quad \quad \quad \text{out} \text{---} 3 \end{array} \supseteq \begin{array}{cc} 1 & 2 \\ \bullet & \bullet \end{array} \supseteq \begin{array}{c} \text{out} \text{---} \text{out} \\ \text{top} \end{array} \right) \\
 p_2 &= \left(\begin{array}{c} \text{top} \text{---} 1 \text{---} 2 \end{array} \supseteq \begin{array}{cc} 1 & 2 \\ \bullet & \bullet \end{array} \supseteq \begin{array}{c} \text{out} \text{---} \text{out} \end{array} \right)
 \end{aligned}$$

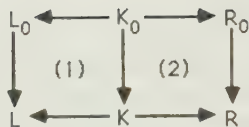
If we add a production which allows to replace *out*-edges by usual (unlabeled) edges, we obtain a grammar which derives (unlabeled) snails.

Hypergraph grammars can be classified according to the form of the productions. This yields a proper language hierarchy. Somewhat surprising, “monotone” hypergraph grammars are more powerful than “context-sensitive” ones. Note that in the string case, context-sensitive and monotone grammars have the same generative power.

1.4 Definition (context-free, context-sensitive, and monotone productions, grammars, and languages)

1. A hypergraph production $(L \supseteq K \subseteq R)$ is called *context-free* if
 - L is a handle, i.e., a hyperedge together with source and target nodes,
 - K is a discrete hypergraph with $V_K = V_L$
 - R is an arbitrary hypergraph with $K \subseteq R$.
2. A hypergraph production $p = (L \supseteq K \subseteq R)$ is called *context-sensitive* if there exists a context-free production $p_0 = (L_0 \supseteq K_0 \subseteq R_0)$, called the *core* of p , and morphisms $L_0 \rightarrow L$, $K_0 \rightarrow K$, and $R_0 \rightarrow R$ such that (1) and (2) below are gluing diagrams (pushouts)⁴.

⁴ For the definition of the category-theoretical notions (pushouts, pullbacks etc.), we refer to Arbib and Manes [AM 75]; for the corresponding set-theoretical descriptions, we refer to [EK 79] and [Eh 79].



3. A hypergraph production $(L \supseteq K \subseteq R)$ is called *monotone* if $\text{size}(L) \leq \text{size}(R)$.
4. A hypergraph grammar is called *context-free* (*context-sensitive*, *monotone*) if all productions are context-free (context-sensitive, monotone).
5. A hypergraph language L is said to be *context-free* (*context-sensitive*, *monotone*) if there is a context-free (context-sensitive, monotone) hypergraph grammar HG generating L . The class of all context-free (context-sensitive, monotone) hypergraph languages is denoted by $CFHL$ ($CSHL$, MHL). The class of all hypergraph languages generated by arbitrary hypergraph grammars is denoted by HL .

Remarks

1. By definition, every context-free hypergraph grammar is context-sensitive, every context-sensitive hypergraph grammar is monotone, and every monotone hypergraph grammar is a hypergraph grammar. Consequently,

$$CFHL \subseteq CSHL \subseteq MHL \subseteq HL.$$

2. Context-free hypergraph grammars correspond to hyperedge-replacement grammars (without empty productions): For a hyperedge-replacement grammar $HRG = (N, T, P, Z)$ (without empty productions) there is an equivalent context-free hypergraph grammar $HG = (N, T, hg(P), Z)$ with $hg(P) = \{hg(p) | p \in P\}$ where, for $p = (A, R')$, $hg(p) = (L \supseteq K \subseteq R)$ denotes the hypergraph production for which L is a *type*(R')-handle with label A , K is the frame of L , and R is the hypergraph $(V_{R'}, E_{R'}, s_{R'}, t_{R'}, l_{R'})$ underlying R' . Conversely, for each context-free hypergraph-grammar $HG = (N, T, P, Z)$, there is an equivalent hyperedge-replacement grammar $HRG = (N, T, hrg(P), Z)$ (without empty productions) with $hrg(P) = \{hrg(p) | p \in P\}$ where, for $p = (L \supseteq K \subseteq R)$, $hrg(p) = (l(L), R')$ denotes the hyperedge-replacement production obtained from p by choosing the label $l(L)$ of the handle L as left-hand side and the multi-pointed hypergraph $(V_R, E_R, s_R, t_R, l_R, s_L(e), t_L(e))$ induced by the R and the relations $L \supseteq K$ and $K \subseteq R$ as right-hand side.

Let us mention that a context-free production $p = (L \supseteq K \subseteq R)$ is applicable to a hypergraph H if and only if there is a hypergraph morphism from L to H . The contact condition is automatically satisfied because all nodes of L are nodes of K . Moreover, the identification condition is satisfied because there is a single hyperedge in L and all nodes of L are nodes of K .

3. For context-sensitive string grammars $CSG = (N, T, P, S)$, there is a corresponding context-sensitive hypergraph grammar $CSHG = (N, T, \hat{P}, S^*)$ where S^* is a string graph with label S , $\hat{P} = \{\hat{p} | p \in P\}$, and, for a production $p = (uAw, uvw)$, $\hat{p}$ denotes the production with core $\hat{p}_0$ induced by the core $p_0 = (A, v)$ of p and context hypergraph induced by the context strings u and w .

4. Using the same methods as in the string case, it can be shown that, for monotone hypergraph grammars, the Membership Problem is decidable.

Concerning the generative power of hypergraph grammars we obtain the following hierarchy where the inclusions are strict:

1.5 Theorem (generative power)

$$CFHL \subset CSHL \subset MHL \subset HL.$$

Proof

1. The string-graph language $\{(a^{n^2})^* | n \geq 1\}$ is in $CSHL$ (see Remark 1.4.3) but not in $CFHL$ according to the Pumping Lemma for hyperedge-replacement grammars (see Chapter IV).
2. The set L of all graphs can be generated by a monotone graph grammar (see Example 1.3). Therefore, L is in MHL . It turns out that it is not in $CSHL$. To show this we assume $L = L(CSHG)$ for some context-sensitive hypergraph grammar $CSHG = (N, T, P, Z)$. Then there exists a context-free hypergraph grammar $CFHG = (N, T, core(P), Z)$ (with $core(P) = \{p_0 | p_0 \text{ is the core of some } p \in P\}$) with $L = L(CSHG) \subseteq L(CFHG)$. Since L is of unbounded connectivity, $L(CFHG)$ is also of unbounded connectivity, a contradiction to the bounded connectivity of hyperedge-replacement languages.
3. Due to a result by Uesu in [Ue 78], HL is the class of all recursively enumerable hypergraph languages. On the other hand, each set in MHL is recursive. Therefore, there exist languages in HL which are not in MHL . $\square$

2. Hypergraph-Replacement Grammars with Application Conditions

Various concepts of graph grammars have been developed in the last 20 years and applied to several areas in computer science and biology ([CER 79], [ENR 83], and [ENRR 87]). In each of these concepts it is defined how and under which conditions graph grammar productions can be applied to a given “mother graph” in order to obtain a derived “daughter graph” (see [KR 88] for a comparative study). The conditions under which productions can be applied are called application conditions. Usually, the application condition is inherent for each specific graph grammar approach and applies in a uniform way to all productions of a graph grammar within this approach. We propose to allow different application conditions for different productions in order to be more flexible using graph grammars for the design of systems in all kinds of application areas. Although the generative power of most of the known general graph grammar concepts is sufficient to generate any recursively enumerable set of graphs it is often convenient to have specific application conditions for each production. Such application conditions, on the one hand, include context conditions like the existence resp. non-existence of certain nodes, edges, or paths in the mother or the daughter graph as well as cardinality restrictions concerning the number of incoming or outgoing edges. On the other hand, they also include embedding restrictions concerning the mappings from the left- resp. right-hand side of the production to the mother resp. the daughter graph. Context conditions in the sense above were informally used in Montanari [Mo 70] and Rosenfeld and Milgram [RM 72] and the use of mappings between graphs, called graph morphisms, is essential for the “Berlin” approach to graph grammars. We combine these concepts leading to a very general but mathematically simple notion of application conditions for productions.

Informally, application of a production $p = (L \supseteq K \subseteq R)$ with application condition to a hypergraph H means:

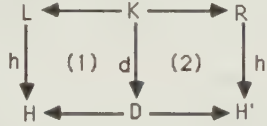
- (1) CHOOSE an occurrence of L in H .
- (2a) CHECK the contact condition and the identification condition.
- (2b) CHECK the application condition.
- (3) REPLACE (the image of) L by R .

Note that the contact condition is a specific application condition, which always has to be satisfied. It guarantees that the result becomes a hypergraph again.

In some cases it is useful to allow “right-hand” application conditions concerning the occurrence of the right-hand side R of the production in the derived hypergraph (“the derived hypergraph has the property *PROP*”). Then we also have to check this condition.

2.1 Definition (application conditions)

1. For a hypergraph X , $\mathcal{MOR}_X$ denotes the set
 $\mathcal{MOR}_X = \{X \rightarrow H \mid H \text{ is a hypergraph and } X \rightarrow H \text{ is a hypergraph morphism}\}.$
2. Given a hypergraph grammar $HG = (N, T, P, Z)$ and a production $p = (L \supseteq K \subseteq R)$ in P , an *application condition* $A(p) = (A_L(p), A_R(p))$ consists of decidable subsets $A_L(p) \subseteq \mathcal{MOR}_L$ and $A_R(p) \subseteq \mathcal{MOR}_R$ of all hypergraph morphisms from L to H resp. R to H' where H and H' are ranging over all $(N \cup T)$ -labeled hypergraphs. $A(p)$ is called *left-sided* if $A_R(p) = \mathcal{MOR}_R$ and *right-sided* if $A_L(p) = \mathcal{MOR}_L$. If $A(p)$ is left-sided [resp. right-sided] we write $A(p) = A_L(p)$ [resp. $A(p) = A_R(p)$].
3. A *hypergraph grammar with application conditions* $HGA = (N, T, P, A, Z)$ consists of a hypergraph grammar $HG = (N, T, P, Z)$, called *underlying hypergraph grammar*, and a set $A = \{A(p) \mid p \in P\}$ of application conditions $A(p)$ for each production $p \in P$.
4. A *direct derivation* in $HGA = (N, T, P, A, Z)$ is a direct derivation $H \Rightarrow H'$ through a production $p = (L \supseteq K \subseteq R)$ in P such that, for the occurring morphisms $L \rightarrow H$ and $R \rightarrow H'$, $(L \rightarrow H) \in A_L(p)$ and $(R \rightarrow H') \in A_R(p)$.



5. The *hypergraph language* $L(HGA)$ induced by HGA consists of all terminal labeled hypergraphs which are derivable in HGA (i.e. with application conditions) from the axiom Z .

Remarks

1. Decidability of the subset $A_L(p)$ [resp. $A_R(p)$] in 2.1 means that there is an algorithm which for each hypergraph morphism $L \rightarrow H$ [resp. $R \rightarrow H'$] decides whether it belongs to $A_L(p)$ [resp. $A_R(p)$] or not. We assume decidability of $A_L(p)$ and $A_R(p)$ in the definition in order to be able to decide for each direct derivation $H \Rightarrow H'$ (through p) in the underlying grammar HG whether it is also a derivation in HGA .
2. By construction, we have $L(HGA) \subseteq L(HG)$ for each grammar HGA with application conditions and underlying grammar HG . Note that the corresponding hypergraph language classes $\mathcal{HCA}$ and $\mathcal{HCL}$ are equal (see Theorem 2.5).
3. Each left-sided application condition $A_L(p)$ induces an *equivalent* right-sided application condition

$$A_L^{-1}(p) = \{R \rightarrow H' \mid \exists H \Rightarrow H' \text{ through } p \text{ with occurrence maps } L \rightarrow H \text{ and } R \rightarrow H' \text{ such that } L \rightarrow H \in A_L(p)\}.$$

Vice versa, each right-sided condition $A_R(p)$ induces an equivalent left-sided condition $A_R^{-1}(p)$. As a consequence, each general application condition $A(p) = (A_L(p), A_R(p))$ can be represented by an equivalent left-sided condition $A_L(p) \cap A_R^{-1}(p)$ or an equivalent right-sided condition $A_R(p) \cap A_L^{-1}(p)$. This shows that for theoretical reasons it is sufficient to use only left-sided or only right-sided conditions. For practical reasons, however, it is convenient to use the general case.

Classification of Application Conditions

Application conditions in general can be classified into *context conditions* concerning the class of admissible hypergraphs and *embedding conditions* concerning set-theoretical properties of hypergraph morphisms. The context conditions may require the existence of some items like nodes, hyperedges, or paths or the non-existence of such items. They are called *positive* in the first and *negative* in the second case. If the items of consideration are included in a given neighborhood of the image of L in H (resp. image of R in H') the conditions may be called *local* and otherwise *global*.

- Typical positive context conditions are the existence of nodes or hyperedges with specific labels in the mother hypergraph H or daughter hypergraph H' .
- Typical negative context conditions are the non-existence of nodes or hyperedges with specific labels, cardinality restrictions (the number of outgoing tentacles is bounded by the constant c), and forbidden structures in the sense of Wankmüller [Wa 83].
- A typical local context condition is the contact condition (see 1.1).
- Typical global context conditions are the existence or non-existence of a cycle in H resp. H' or a path between nodes in the image of L resp. R .
- Typical embedding conditions are the injectivity of $L \rightarrow H$ resp. $R \rightarrow H'$ and the identification condition (see 1.1).

The *gluing condition*, which is a conjunction of the contact condition and the identification condition (see 1.1), is an application condition which is satisfied if and only if there is a direct derivation in the underlying grammar HG (see [Eh 79]). Hence, it is not necessary to require one of these conditions separately in the set A of application conditions of a grammar HGA .

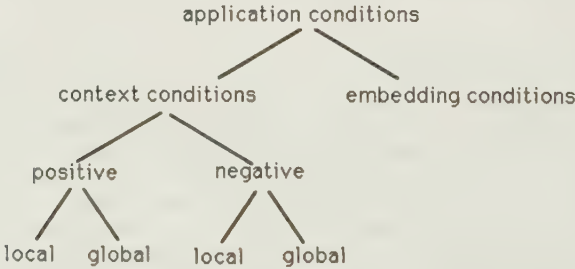


Figure 2.1: Classification of Application Conditions.

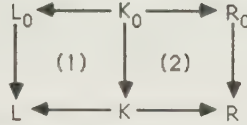
A local positive context condition can be explicitly included into the production. But it can also be formulated as a concrete context condition for the production p_0 . Vice versa, given a production p_0 and a concrete context, we can construct a production p by “gluing” of p_0 with the concrete context.

2.2 Definition (concrete context condition and context enlargement)

1. A (left-sided) application condition $A_L(p_0)$ for a production $p_0 = (L_0 \supseteq K_0 \subseteq R_0)$ is called *concrete context condition* for p_0 if there is a *concrete context hypergraph* K and an injective hypergraph morphism $K_0 \rightarrow K$ such that

$$A_L(p_0) = \{L_0 \rightarrow H \mid \exists K \rightarrow H \text{ such that diagram } K_0 K L_0 H \text{ is a weak pullback}\}^1$$

2. The hypergraph production $p = (L \supseteq K \subseteq R)$ is called a *context enlargement* of p_0 via $K_0 \rightarrow K$.



2.3 Lemma

For each direct derivation $H \Rightarrow H'$ through a production p_0 with concrete context condition defined by $K_0 \rightarrow K$ there is a corresponding direct derivation $H \Rightarrow H'$ through p , where p is the context enlargement of p_0 via $K_0 \rightarrow K$, and vice versa.

Proof

The proof is given in [EH 86]. □

Lemma 2.3 can be used to show the equivalence of context-free hypergraph grammars with concrete context conditions and context-sensitive hypergraph grammars as defined in Section 1.

2.4 Theorem

Context-free hypergraph grammars with concrete context conditions are equivalent to context-sensitive hypergraph grammars without application conditions, i.e., the corresponding hypergraph languages and classes of hypergraph languages $\mathcal{CFH\mathcal{L}A}$ and $\mathcal{CS\mathcal{H}L}$ are equal.

Proof

Since context-sensitive productions are exactly context enlargements of context-free productions in the sense of 2.2, Theorem 2.4 follows immediately from Lemma 2.3. □

¹ The diagram $K_0 K L_0 H$ is called a weak pullback if it is commutative, i.e., $K_0 \rightarrow K \rightarrow H = K_0 \rightarrow L_0 \rightarrow H$, and for all $x \in L_0$ with $(L_0 \rightarrow H)(x) \in (K \rightarrow H)(K)$, $x \in K_0$. This means that the overlapping of L_0 and K in H is only possible in nodes of K_0 . The condition implies the existence of a morphism $L \rightarrow H$ such that $L_0 \rightarrow L \rightarrow H = L_0 \rightarrow H$ and $K \rightarrow L \rightarrow H = K \rightarrow H$.

Concerning the generative power of hypergraph grammars with application conditions, we can show the following result:

2.5 Theorem (generative power)

Let $\mathcal{CFHLC}$ be the class of languages generated by context-free hypergraph grammars with concrete context conditions and $\mathcal{HLC}$ be the class of languages generated by hypergraph grammars with application conditions. Then

$$\mathcal{CFHL} \subset \mathcal{CFHLC} \subset \mathcal{HL} \quad \text{and} \quad \mathcal{HL} = \mathcal{HLC}.$$

Proof

The string-graph language $\{(a^n)^* \mid n \geq 1\}$ is in $\mathcal{CSHL}$ and, by Theorem 2.4, in $\mathcal{CFHLC}$ but not in $\mathcal{CFHL}$. The set of all complete graphs is in $\mathcal{HL}$, but not in $\mathcal{CSHL}$ (see Section 1). By Theorem 2.4, it is not in $\mathcal{CFHLC}$. By definition, $\mathcal{HL} \subseteq \mathcal{HLC}$. Due to a result by Uesu in [Ue 78], $\mathcal{HL}$ is the class of all recursively enumerable hypergraph languages. Since all application conditions are decidable, $\mathcal{HLC}$ cannot include any other language. $\square$

2.6 Final Remarks

1. Using hypergraph grammars without application conditions as basic model, the only way to incorporate context conditions is to include the concrete context into the left- and right-hand sides of the productions. The presented version of application conditions is much more general. In particular, it includes negative context conditions in the sense of Gottlieb and Furtado [GF 74] and can be seen as a formalization of application conditions for graph grammar productions which were given in Montanari [Mo 70] and Rosenfeld and Milgram [RM 72].
2. Although the notion of application conditions is mathematically precise it still allows all kinds of mathematical formalisms to define specific subclasses. We only have required decidability, but we propose to use the methods from formal language theory, logic, algebra, or graph theory in order to define specific subclasses according to the application in mind. It is interesting to study in addition to concrete context conditions also other special cases of application conditions explicitly.
3. There are still some open questions concerning the generative power, e.g., the following: Are context-free hypergraph grammars with general application conditions more powerful than those with concrete context conditions which are equivalent to context-sensitive hypergraph grammars?
4. Let us mention that programmed graph grammars in the sense of Bunke [Bu 79] provide also a way to restrict applications of productions. While the application conditions are concerning single productions, programmed graph grammars are restricting possible sequences of productions to be applied in a derivation sequence by suitable control graphs. It depends on the kind of application which method should be taken and it may also be useful to combine both of them.

3. Parallel Hyperedge–Replacement Grammars

In this section, parallel hyperedge–replacement grammars are introduced. They may be seen as a hypergraph version of ETOL systems (see, e.g. Herman and Rozenberg [HR 75]). This kind of systems is based on simultaneously context-free rewriting. Moreover, in a ETOL-system, the production set is divided into subsets, called *tables*. At each derivation step of the rewriting process, only productions belonging to the same table can be used. The underlying biological motivation is that different sets of rules may be needed at different developmental stages. Let us mention that, additionally, the set of symbols is divided into a set of nonterminal symbols and a set of terminal symbols.

3.1 Definition (parallel hyperedge–replacement grammars)

1. A *parallel hyperedge–replacement grammar* is a system $PHRG = (N, T, \mathcal{P}, Z)$, where N and T are finite sets of labels, $\mathcal{P}$ is a finite set of finite production sets over N (called set of *tables*) and $Z \in \mathcal{H}_N$ is the axiom.
2. The language $L(PHRG)$ generated by $PHRG$ consists of all terminal hypergraphs which can be derived from Z if in each derivation step all hyperedges are replaced applying productions of one production set P of $\mathcal{P}$:

$$L(PHRG) = \{H \in \mathcal{H}_T \mid Z \xrightarrow[\mathcal{P}]{\Rightarrow} H\}^1.$$

Remarks

1. Parallel hyperedge–replacement grammars are closely related to ETOL systems (see, e.g., Herman and Rozenberg [HR 75]).
2. For a parallel grammar $PHRG = (N, T, \mathcal{P}, Z)$ it may be required that for each $P \in \mathcal{P}$ and each $A \in N$ there is at least one production $p \in P$ with $lhs(p) = A$.

The relationship between parallel hyperedge–replacement grammars and usual hyperedge–replacement grammars is formulated in Theorem 3.2.

3.2 Theorem

1. Let $HRG = (N, T, P, Z)$ be a usual hyperedge–replacement grammar. Then the parallel hyperedge–replacement grammar $PHRG = (N, T, \{P\}, Z)$ generates the same language as HRG , i.e., $L(HRG) = L(PHRG)$.
2. Let $PHRG = (N, T, \mathcal{P}, Z)$ be a parallel hyperedge–replacement grammar. Then the usual hyperedge–replacement grammar $HRG = (N, T, \bigcup_{P \in \mathcal{P}} P, Z)$ generates a language including the language of $PHRG$, i.e., $L(PHRG) \subseteq L(HRG)$.

¹ Here the following notations are used: Given a direct derivation $H \xrightarrow[\mathcal{P}]{\Rightarrow} H'$ through the “full” base $prod : E_H \rightarrow P$, we write $H \xrightarrow[\mathcal{P}]{\Rightarrow} F'$ provided that the fullness is of interest. If $H_0 \xrightarrow[\mathcal{P}_1]{\Rightarrow} H_1 \xrightarrow[\mathcal{P}_2]{\Rightarrow} \dots \xrightarrow[\mathcal{P}_k]{\Rightarrow} H_k$ is a derivation from H_0 to H_k and $P_1, P_2, \dots, P_k \in \mathcal{P}$, then we write $H_0 \xrightarrow[\mathcal{P}]{\Rightarrow} H_k$ or $H_0 \xrightarrow[\mathcal{P}]{\Rightarrow} H_k$ provided that $\mathcal{P}$ is of interest.

Proof

The first statement follows immediately from the existence of canonical derivations and the possibility for sequentializing parallel derivations (see, e.g., [Kr 87b]). The second statement follows from the fact that each derivation in *PHRG* is a parallel derivation in *HRG*. □

3.3 Examples

1. The set of all graphs (snowflakes) underlying patterns of the form

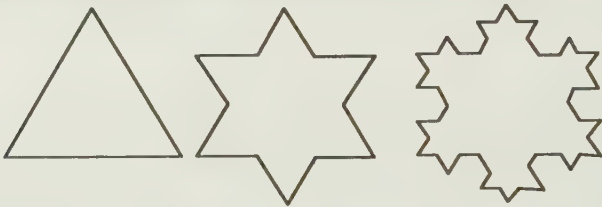
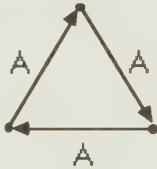
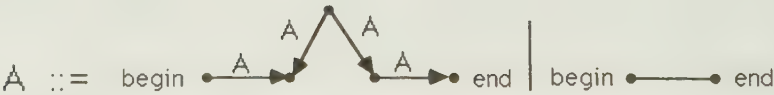


Figure 3.1: Snowflakes.

can be generated by the parallel (hyper)edge-replacement grammar $PERG_{\text{SNOW}}$ with the axiom



and the productions



where the production set is divided into the set containing the “generating” production and the set containing the “terminating” production.

2. The set of all graphs underlying patterns of the form

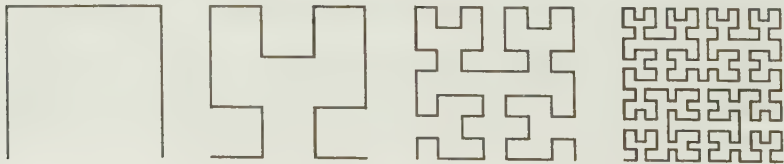
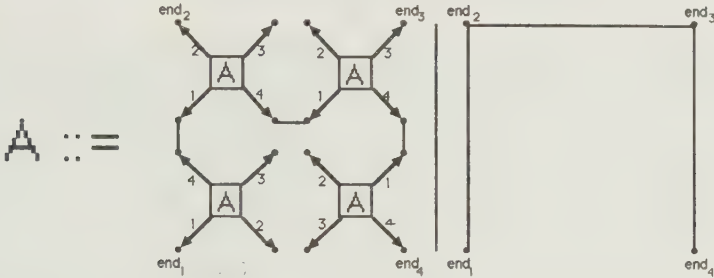


Figure 3.2: Hilbert patterns.

can be generated by the parallel hyperedge-replacement grammar $PHRG_{\text{Hilbert}}$ where the axiom is a $(0, 4)$ -handle with label A , the productions are of the form



and the production set is divided into the set with the “generating” production and the set with the “terminating” production.

3.4 Theorem (generative power)

Let $PHRL$ be the class of languages generated by parallel hyperedge-replacement grammars. Then

$$HRL \subset PHRL.$$

Proof

By Theorem 3.2, $HRL \subseteq PHRL$. As shown in Example 3.3, the set of all snowflakes is in $PHRL$. By the Pumping Lemma resp. the corollary concerning the semilinear growing (see Chapter IV), it can be shown that the set of snowflakes is not in HRL . $\square$

4. Figure-Generating Grammars Based on Hyperedge Replacement

The systematic generation and manipulation of artificial pictures and patterns is one of the most challenging tasks of computer science. Among the various attempts, syntactic proposals, where classes of patterns are described by grammars, are found in the literature (see, e.g., Feder [Fe 68+71], Shaw [Sh 69], Fu [Fu 74], Pfaltz and Rosenfeld [PR 69], Bunke [Bu 83], de Does and Lindenmayer [DL 83], Prusinkiewicz [Pr 86]).

In this section, we discuss an approach to pattern generation based on hyperedge-replacement grammars. Hyperedge-replacement grammars provide a simple, well-studied method for generating graphs and hypergraphs. For pattern generation — instead of (hyper)graph generation — the productions and the initial (hyper)graph of hyperedge-replacement grammars are equipped with layout information so that the generated graphs and hypergraphs can easily be displayed as 2-dimensional (or 3-dimensional) patterns on a “screen”, which is given by the layout of the initial (hyper)graph. In this way, the graph-grammar approach of hyperedge-replacement grammars can be utilized for the production of surprisingly rich and pretty patterns, which may be considered as kinds of collages combined of bits of pictures (as opposed to patterns drawn and computed pixel by pixel).

In the following, we consider 2-dimensional line figures which are equipped with hyperedges. Roughly speaking, a figure consists of points and lines in the plane. It may be equipped by some labeled hyperedges (objects which may have an arbitrary number of “tentacles” to grip at points). The hyperedges grip at some points and determine regions for other figures. In this sense hyperedges serve as place-holders for figures.

4.1 Definition (figures)

1. Let N be an arbitrary, but fixed alphabet, called set of *labels*. A *figure* over N is a system $F = (V, L, E, \text{nod}, \text{lab}, \text{ext})$ where V is a finite set of *points* in the plane, L is a finite set of *lines* in the plane, where each line connects two, not necessarily different, points of V , E is a finite set of *hyperedges*, $\text{nod} : E \rightarrow V^*$ is a mapping assigning a sequence of points $\text{nod}(e)$ to each $e \in E$, $\text{lab} : E \rightarrow N$ is a mapping assigning a *label* to each $e \in E$, and $\text{ext} \in V^*$ is a sequence of *external points* of F such that the following properties are satisfied:

(1) For each $e \in E$, $\text{nod}(e)$ determines a polygon such that $(\text{nod}(e)) \cap V = \emptyset$, $(\text{nod}(e)) \cap l = \emptyset$ for all $l \in L$, and $(\text{nod}(e)) \cap (\text{nod}(e')) = \emptyset$ for all $e' \in E$ with $e \neq e'$.¹
(2) ext determines a polygon such that $V \subseteq [\text{ext}]$, $l \subseteq [\text{ext}]$ for all $l \in L$, and $[\text{nod}(e)] \subseteq [\text{ext}]$ for all $e \in E$.

2. The components of F are denoted by V_F , L_F , E_F , nod_F , lab_F , ext_F respectively.

¹ Let $p_1, \dots, p_n$ be points in the plane. Then the sequence $p_1 \dots p_n$ determines a polygon provided that the line segments $\overline{p_1 p_2}, \overline{p_2 p_3}, \dots, \overline{p_n p_1}$ form a polygon in the usual sense. The polygon is denoted by $p_1 \dots p_n$. $(p_1 \dots p_n)$ denotes the interior of the polygon, i.e., the set of all points that are inside the polygon; $[p_1 \dots p_n]$ denotes the set of all points which are inside or on a side of the polygon.

3. The set of all figures over N is denoted by $\mathcal{F}_N$; the set of all figures $F \in \mathcal{F}_N$ without hyperedges (with $E_F = \emptyset$) is denoted by $\mathcal{F}$.

Remarks

1. A figure is simply a line drawing placed on a polygon in the plane. It may contain some hyperedges additionally. Each hyperedge possesses many “tentacles” to grip at points and to determine a polygon. By condition (1), the interior of the polygons is not occupied by points, lines or other hyperedges, i.e., no point lies in the interior, no line crosses the interior and no other hyperedge requires parts of the interior of the polygon of a hyperedge. In this sense, the region determined by a hyperedge may be seen as region kept free by the hyperedge. (Note that only the interior of the polygons is kept free; some points may stay on a side of the polygon, lines may touch parts of the sides, and different hyperedges may “meet” each other in common sides).

2. Hyperedges are equipped with additional information; each hyperedge is associated with a hyperedge label. The hyperedge labels provide a control of hyperedge replacement: While a hyperedge may be seen as a place-holder for figures, their label may restrict the set of replacing figures.

3. Beside points, lines and hyperedges, there are some distinguished points, called external points, pointing out the “frame” of a figure: By condition (2), all points, lines and regions of hyperedges stay in the region of the figure (determined by the external points of the figure). The explicit declaration of the frame will be useful if we will replace a hyperedge in a figure by a “fitting” replacing figure.

4. In a similar way, more general types of figures, e.g., figures of points, lines and areas, colored figures, higher-dimensional figures etc., may be handled.

5. Note, that there is a close relationship between figures and multi-pointed hypergraphs: On the one hand, each figure F over N is associated with a multi-pointed (undirected, hyperedge-labeled) hypergraph, called *underlying multi-pointed hypergraph*. This hypergraph is given by a system $(V_F, L_F + E_F, \text{nod}, \text{lab}, \text{ext}_F)$ where V_F is the set of *nodes* (or vertices), $L_F + E_F$ is the set of *edges* and *hyperedges*, $\text{nod} : E_F + L_F \rightarrow V_F^*$, $\text{lab} : E_F + L_F \rightarrow N + \{\star\}$ are extensions of nod_F and lab_F , respectively, such that, for all $l \in L_F$, nod assigns (a sequence of) nodes associated to the end-points of l and lab assigns the invisible label (indicated by $\star$) to l . On the other hand, a lot of multi-pointed (undirected, hyperedge-labeled) hypergraphs over $N + \{\star\}$ (whose ordinary edges are labeled by $\star$ and whose hyperedges with more than two tentacles are labeled in N) may be equipped with layout information such that we get a figure over N .

4.2 Example

1. The plane graphs shown in Figure 4.1 may be seen as figures in the sense of Definition 4.1 provided that the points $\text{ext}_1, \dots, \text{ext}_6$ (resp. $\text{ext}'_1, \dots, \text{ext}'_6$) are distinguished as external points. Note that the lines of a figure may be straight or curved lines.² Figures may be equipped with hyperedges, as in Figure 4.1(a). These hyperedges reserve regions in the plane for other figures.

² A graph which is actually drawn in the plane so that its lines intersect only at points of the graph is said to be a *plane graph* (see, e.g., Anderson [An 70]).

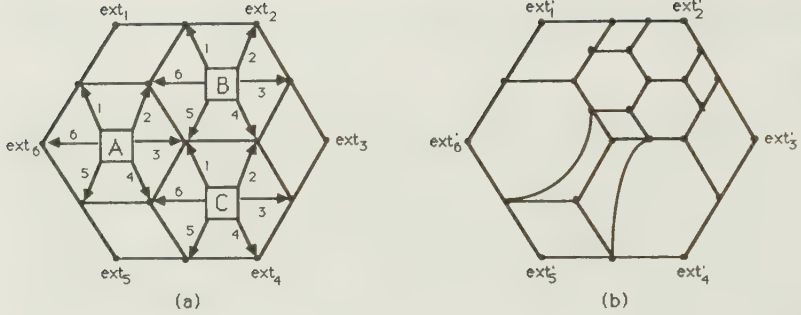


Figure 4.1: Figures.

4.3 Definition (topological equivalence and equality of figures)

1. Let $F, F' \in \mathcal{F}_N$. Then F and F' are *topologically equivalent*, denoted by $F \triangle F'$, if there are bijective mappings $f_V : V_F \rightarrow V_{F'}$, $f_L : L_F \rightarrow L_{F'}$, and $f_E : E_F \rightarrow E_{F'}$ such that $f_V(\text{end}_F(l)) = \text{end}_{F'}(f_L(l))$ for all $l \in L_F$ ³, $f_V^*(\text{nod}_F(e)) = \text{nod}_{F'}(f_E(e))$ and $\text{lab}_F(e) = \text{lab}_{F'}(f_E(e))$ for all $e \in E_F$, $f_V^*(\text{ext}_F) = \text{ext}_{F'}$.
2. Let $F, F' \in \mathcal{F}_N$. Then F and F' are *equal (up to isomorphism)* if $V_F = V_{F'}$, $L_F = L_{F'}$, $\text{ext}_F = \text{ext}_{F'}$, and there is a bijective mapping $\text{iso} : E_F \rightarrow E_{F'}$ such that $\text{nod}_F(e) = \text{nod}_{F'}(\text{iso}(e))$ and $\text{lab}_F(e) = \text{lab}_{F'}(\text{iso}(e))$ for all $e \in E_F$.

Hyperedges in figures serve as place-holders for figures. Hence the key construction is the replacement of some hyperedges in a figure by figures. While a hyperedge grips to its points according to our conventions, a figure is equipped with the information which of its points may be seen as external points. If the points of the hyperedge correspond to the external points of a figure, the hyperedge may be replaced by this figure. It turns out that replacing a hyperedge by a “fitting” figure yields a figure. The following formal definition allows to replace an arbitrary collection of hyperedges in parallel.

4.4 Definition (hyperedge replacement)

1. Let $F \in \mathcal{F}_N$ and $e \in E_F$. Then $\text{Fig}(e)$ denotes the set of all figures “fitting” for e , i.e., $\text{Fig}(e) = \{R \in \mathcal{F}_N | \text{ext}_R = \text{nod}_F(e)\}$.⁴
2. Let $F \in \mathcal{F}_N$ and $B \subseteq E_F$. Let $\text{repl} : B \rightarrow \mathcal{F}_N$ be a mapping with $\text{repl}(b) \in \text{Fig}(b)$ for all $b \in B$. Then the *replacement* of B in F through repl yields the figure X given by

- $V_X = V_F \cup \bigcup_{b \in B} V_{\text{repl}(b)}$,
- $L_X = L_F \cup \bigcup_{b \in B} L_{\text{repl}(b)}$,

³ For $F \in \mathcal{F}_N$ and $l \in L_F$, $\text{end}_F(l)$ denotes the set of endpoints of l connected by l .

⁴ The symbol “=” denotes the equality of two strings.

- $E_X = (E_F - B) + \sum_{b \in B} E_{\tau_{\mathcal{F}}(b)}$,
 - each hyperedge keeps its points and its label,
 - $ext_X = ext_F$.
3. X will be denoted by $REPLACE(F, repl)$.

Remark

The replacement of hyperedges in figures by fitting replacing figures yields figures again: Let $F \in \mathcal{F}_N$, $B \subseteq E_F$, and $repl : B \rightarrow \mathcal{F}_N$ be a mapping with $repl(b) \in Fig(b)$ for all $b \in B$. Then $REPLACE(F, repl) \in \mathcal{F}_N$.

In the following we introduce figure-generating hyperedge-replacement grammars closely related to hyperedge-replacement grammars and context-free string grammars. They can be seen as a figure-manipulating and figure-language generating device.

Based on hyperedge replacement, one can derive figures from figures by applying productions of a simple form. A production is formed of a label $A \in N$ and a concrete figure $R \in \mathcal{F}_N$. It may be applied to a hyperedge e with label A provided that R fits for e . More generally, we allow to apply the production to a hyperedge e with label A provided that R can be “transformed” to a figure R' fitting for e . In this case, the hyperedge e is replaced by the fitting figure R' . In this sense, a production may be seen as a scheme for replacements.

Various kinds of transformations ⁵ may be considered, e.g.,

- identical transformations,
- similarity transformations,
- affine transformations,
- topological transformations.

Depending on the kind of transformations allowed, one may derive less or more figures from figures by applying productions. Hence, we have to indicate what kind of transformations are considered. Note that it seems to be suitable to restrict on transformations transforming figures into topological equivalent ones.

4.5 Definition (productions and derivations)

Let $\sim$ be a fixed equivalence relation on $\mathcal{F}_N$ with $\sim \subseteq \Delta$. ⁶

1. A *production* over N is an ordered pair $p = (A, R)$ with $A \in N$ and $R \in \mathcal{F}_N$. A is called the *left-hand side* of p and is denoted by $lhs(p)$, R is called the *right-hand side* of p and is denoted by $rhs(p)$.
2. Let P be a fixed set of productions over N . Let $F \in \mathcal{F}_N$ and $B \subseteq E_F$. A mapping $prod : B \rightarrow P \times \mathcal{F}_N$ is called a *base* in F (with respect to $\sim$) if $lhs(prod_i(e)) = lab_F(e)$, $rhs(prod_i(e)) \sim prod_i(e)$, and $prod_i(e) \in Fig(e)$ for all $e \in B$. ⁷

⁵ With respect to geometrical transformations we refer to Gans [Ga 69].

⁶ Δ denotes the topological equivalence (see Def. 4.3).

⁷ The i th component of $prod(e)$ is denoted by $prod_i(e)$ ($i = 1, 2$).

3. Let $F, F' \in \mathcal{F}_N$, let $prod : B \rightarrow P \times \mathcal{F}_N$ be a base in F (w.r.t. $\sim$). Then F *directly derives* F' through $prod$ (w.r.t. $\sim$) if $F' = REPLACE(F, prod_2)$ (up to isomorphism). We write $F \xRightarrow{P} F'$ or $F \Rightarrow F'$.

4. A sequence of direct derivations (w.r.t. $\sim$) $F_0 \Rightarrow F_1 \Rightarrow \dots \Rightarrow F_k$ is called a *derivation* from F_0 to F_k (with respect to $\sim$). It is abbreviated by $F_0 \xRightarrow{*} F_k$ or $F_0 \xRightarrow{P} F_k$ provided that the set P of productions is of interest.

Remark

The application of a production $p = (A, R)$ to a figure F (with respect to $\sim$) requires the following steps only:

- (1) Choose a hyperedge $e \in E_F$ with label A .
- (2) Transform the figure R to a fitting figure R' with $R' \sim R$ (if possible).
- (3) Replace the hyperedge e in F by the transformed figure R' .

Using the introduced concepts of productions and derivations figure-generating hyperedge-replacement grammars and figure languages can be introduced in a straightforward way.

4.6 Definition (figure-generating grammars and languages)

1. A *figure-generating (hyperedge-replacement) grammar*, short figure-generating grammar, is a system $FG = (N, P, Z)$ where N is a set of labels, P is a finite set of productions over N , and $Z \in \mathcal{F}_N$ is a figure, called *axiom*.
2. The *figure language* $L_{\sim}(FG)$ *generated by* FG (with respect to $\sim$) consists of all figures without hyperedges which can be derived from Z applying productions of P :

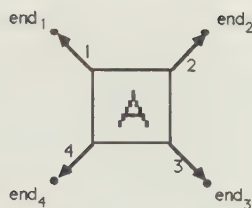
$$L_{\sim}(FG) = \{F \in \mathcal{F} \mid Z \xRightarrow{*} F\}.$$

Remarks

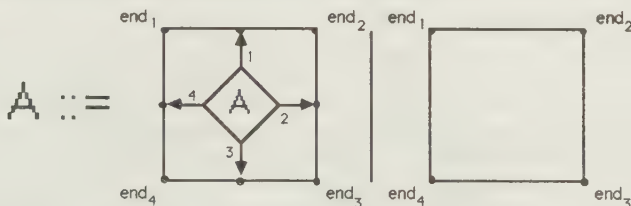
1. Grammars as introduced above are based on the replacement of hyperedges. The objects generated by this kind of grammars are figures in the sense of Def. 4.1. To emphasize these aspects, they are called figure-generating (hyperedge-replacement) grammars.
2. Given a figure-generating grammar, the axiom plays an essential role: it determines the “frame” for all generated figures. In particular, a language containing figures with different frames (the frame of a figure is given by its sequence of external nodes) cannot be generated by grammars introduced above.

4.7 Examples

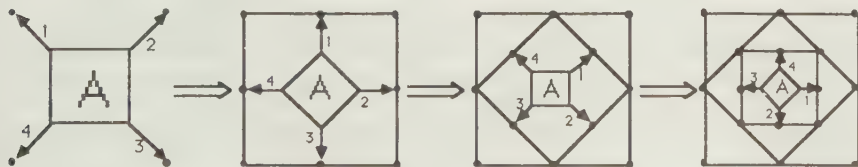
1. Let us consider the figure-generating grammar $FG = (\{A\}, P, Z)$ where Z denotes the figure



and P consists of two productions given in a kind of Backus-Naur-form:



Starting from the figure Z , the first production can be applied to the single hyperedge in Z provided that we allow the translation of the right-hand side to the region of the hyperedge. Obviously, the resulting figure again possesses only one hyperedge which is labeled by A . Now the first production (resp. the second production) may be applied provided that it is allowed to contract, rotate and translate the right-hand side. Allowing similarity transformations only, a derivation in the figure-generating grammar has the following shape:

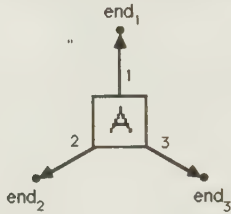


The figures generated by FG (with respect to similarity) are of the form

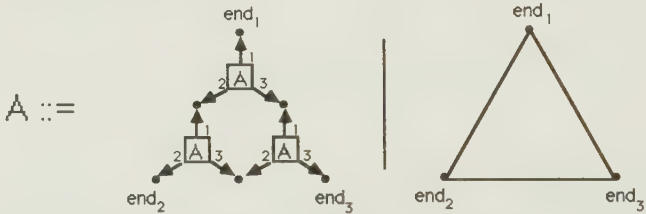


Figure 4.2: Inscribed squares.

2. The figure-generating grammar $FG = (\{A\}, P, Z)$ with the axiom



and the productions p_1 and p_2 given by



allows to generate highly regular figures, called *Sierpinski triangles*.

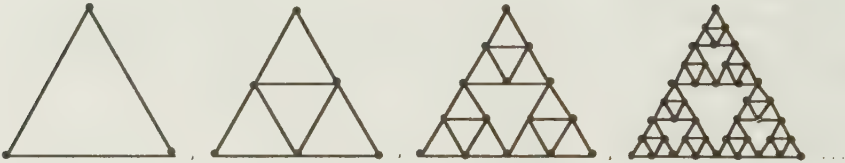
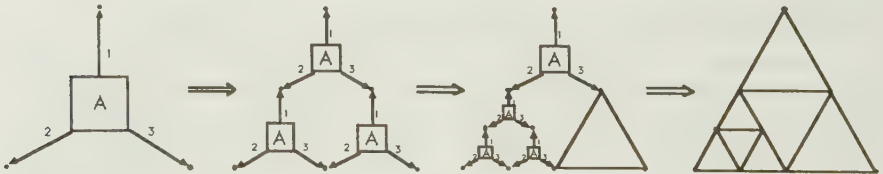


Figure 4.3: Sierpinski triangles.

Furthermore, there are several other figures derivable from the axiom as the following derivation shows.



Note that in the second derivation step of the derivation only two of three hyperedges are replaced. Moreover, different productions are used in this derivation step. If one is interested in generating the Sierpinski triangles only, then one has to use the ETOL-mode of rewriting, i.e. for this example,

- (1) Replace all hyperedges in each step.
- (2) Use one of the productions only in each step.

Therefore, it is important to consider parallel figure-generating hyperedge-replacement grammars beside the usual ones. In parallel figure-generating grammars, we have sets of production sets; in each derivation step we choose one set and replace all hyperedges by applying productions of this set. (A formal definition is given in [HK 88].) With respect to our example, the parallel figure-generating grammar $PFG = (\{A\}, \{\{p_1\}, \{p_2\}\}, Z)$ in which the “generating” production p_1 and the “terminating” production p_2 are in different production sets generates only the Sierpinski triangles.

4.8 Final Remarks

1. Several results known for hyperedge-replacement grammars can be adapted to figure languages generated by figure-generating hyperedge-replacement grammars. In [HK 88], it is shown that the Context-Freeness Lemma, the Fixed-Point Theorem, as well as the Pumping Lemma can be formulated for figure-generating grammars. The Fixed-Point Theorem provides an alternative mechanism for generating figure languages described by figure-generating grammars. The Pumping Lemma provides information on the structure of complex figures generated by a grammar and may be used to show that certain kinds of figure languages cannot be generated by a figure-generating hyperedge-replacement grammar.

2. The approach for figure generation based on hyperedge-replacement grammars seems to be attractive in two respects: It covers various interesting examples and it promises the prospect of rich results. But more work has to be done to get a clearer picture of figure-generating grammars.

(1) Beside the figure-generating hyperedge-replacement grammar, the kind of transformations for adapting a figure to the region of a hyperedge plays an important role. We have concentrated our considerations to affine transformations. Which other transformations will work? How does the restriction (resp. generalization) of the used transformation concept influence the generative power?

(2) The figures, considered in this section, are of a simple form. More general types of figures seem to be of interest, e.g., figures of points, lines and areas, colored figures, higher-dimensional figures, etc. . Which type of figures should be investigated?

(3) Another interesting type of extension concerns the moving of points and lines after they have been generated. Up to now, all points and lines keep their place in the plane.

3. Figure-generating hyperedge-replacement grammars can be extended to parallel figure-generating hyperedge-replacement grammars in order to obtain more expressive power. A formal definition can be found in [HK 88].

4. We are dreaming of the specification of hyperedge-replacement grammars generating pictorial illusions of clouds, mountains, forests, and of whole landscapes in the style of Mandelbrot's fractals (see, e.g., [Ma 77]). It is obvious that the approach is far from this stage of development. Our concept is lacking, until now, a mechanism which disguises the regularity of the generated pictures.

5. Bibliographic Note

Hypergraph grammars were used in [HKP 88] to describe the evaluation of functional expressions, and in particular, of algebraically specified operations. Graph grammars with application conditions were investigated in [EH 86]. Figure-generating grammars and parallel figure-generating grammars were introduced in [HK 88].

Conclusion

The theory of graph grammars forms a well-motivated area of theoretical computer science. It has originally attempted to attack some fundamental problems in the area of picture processing (see, e.g., Rosenfeld and Milgram [RM 72] and Pfaltz [Pf 72]) and subsequently the development of the theory was motivated (and influenced) by various application areas such as databases, incremental compilers, data flow analysis, operational semantics, abstract data types, data structures and developmental biology (see, e.g., [Eh 79], [Na 79]). Due to the diversity of motivations the theory of graph grammars considers a very wide spectrum of graph rewriting systems (see, e.g., [CER 79], [ENR 83], and [ENRR 87]), where rather different kinds of questions are posed for different models. Thus, although by now some models possess a quite well-developed mathematical theory (see, e.g., [Eh 79], [JR 80 a+b], [Ja 83]) or a successful range of applications (see, e.g., [Na 79], [CER 79], [ENR 83], and [ENRR 87]), in general the theory of graph grammars would benefit from more "coherence".

The theory of hyperedge-replacement grammars, the basic model considered in this thesis, is an attempt to build up systematically a theory for manipulating highly structured objects. The goals of these investigations are twofold: On the one hand, to build up a formal theory based on "simple and natural" assumptions, on the other hand, to propose one model of graph grammars for a variety of diverse types of problems. The work may be seen as part of a systematic investigation of hyperedge-replacement grammars as well as a counterpart to the investigations on node-label controlled graph grammars considered by Dirk Janssens and Grzegorz Rozenberg (see, e.g., [JR 80a+b], [Ja 83]).

This thesis consists mainly of two types of investigation:

Firstly, problems with respect to the generating processes and generative power of hyperedge-replacement grammars. In Chapter III, a Kleene-Type Characterization and a Fixed-Point Theorem are stated; they provide alternative mechanisms for generating hyperedge-replacement languages. In Chapter IV, a Pumping Lemma for hyperedge-replacement languages is stated and used to prove results on the generative power of hyperedge-replacement grammars. In particular, it is shown that their generative power depends strongly on the so-called order, which is an upper bound for the number of "tentacles" a non-terminal hyperedge may have. The order induces an infinite hierarchy of classes of hypergraph languages, which remains infinite even if one considers graph or string(-graph) languages.

Secondly, some decidability problems with respect to hyperedge-replacement languages are investigated, e.g. in Chapter V, questions of the form

- (1) Is there a hypergraph in the generated language satisfying the property P ?
- (2) Do all hypergraphs in the generated language satisfy the property P ?
- (3) Are the values of all hypergraphs in the generated language bounded?

are shown to be decidable for a class of hyperedge-replacement grammars if the corresponding graph-theoretic property P is "compatible" resp. if the corresponding quantity function f is "compatible" with the derivation process of the given grammars and is built up by maxima, sums and products.

References

- [AU 72] A.V. Aho, J.D. Ullman: The Theory of Parsing, Translation, and Compiling, Vol. I: Parsing, Prentice-Hall, Englewood-Cliffs, N.J., 1972
- [An 70] S. Anderson: Graph Theory and Finite Combinatorics, Markham, Chicago 1970
- [AM 75] M.A. Arbib, E.G. Manes: Arrows, Structures, and Functors, Academic Press, New York, 1975
- [BPS 61] Y. Bar-Hillel, M. Perles, E. Shamir: On Formal Properties of Simple Phrase-Structure Grammars, Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung 14, 143-177, 1961
- [BC 87] M. Bauderon, B. Courcelle: Graph Expressions and Graph Rewriting, Math. Systems Theory 20, 83-127, 1987
- [Br 83] F.J. Brandenburg: On the Complexity of the Membership Problem of Graph Grammars, Proc. Graphtheoretic Concepts in Computer Science 1983, Trauner Verlag, Linz, 40-49, 1983
- [Bu 79] H. Bunke: Programmed Graph Grammars, Lect. Not. Comp. Sci. 73, 155-166, 1979
- [Bu 83] H. Bunke: Graph Grammars as a Generative Tool in Image Understanding, Lect. Not. Comp. Sci. 153, 8-19, 1983
- [CER 79] V. Claus, H. Ehrig, G. Rozenberg (eds.): Graph-Grammars and Their Application to Computer Science and Biology, Lect. Not. Comp. Sci. 73, 1979
- [Co 87a] B. Courcelle: A Representation of Graphs by Algebraic Expressions and its Use for Graph Rewriting Systems, Lect. Not. Comp. Sci. 291, 112-132, 1987
- [Co 87b] B. Courcelle: On Context-Free Sets of Graphs and Their Monadic Second-Order Theory, Lect. Not. Comp. Sci. 291, 133-146, 1987
- [Co 88] B. Courcelle: An Axiomatic Definition of Context-Free Rewriting and its Application to NLC Grammars, Symp. Theor. Aspects of Comp. Sci. 1988 (STACS'88), Lect. Not. Comp. Sci. 294, 237-247, 1988
- [DL 83] M. de Does, A. Lindenmayer: Algorithms for the Generation and Drawing of Maps Representing Cell Clones, Lect. Not. Comp. Sci. 153, 39-58, 1983
- [DG 78] P. Della Vigna, C. Ghezzi: Context-Free Graph Grammars, Inf. Contr. 37, 207-233, 1978
- [Eh 79] H. Ehrig: Introduction to the Algebraic Theory of Graph Grammars, Lect. Not. Comp. Sci. 73, 1-69, 1979

- [EH 86] H. Ehrig, A. Habel: Graph Grammars with Application Conditions, in: G. Rozenberg, A. Salomaa (eds.): *The Book of L*, Springer-Verlag, Berlin, 87-100, 1986
- [EK 75] H. Ehrig, H.-J. Kreowski: Church-Rosser Theorems Leading to Parallel and Canonical Derivations in Graph Grammars, TU Berlin, Comp. Sci. Report 75-27, 1975
- [EK 79] H. Ehrig, H.-J. Kreowski: Pushout-Properties: An Analysis of Gluing Constructions for Graphs, *Math. Nachr.* 91, 135-149, 1979
- [EKMRW 81] H. Ehrig, H.-J. Kreowski, A. Maggiolo-Schettini, B.K. Rosen, J. Winkowski: Transformations of Structures: An Algebraic Approach, *Math. Syst. Theory* 14, 305-334, 1981
- [ENR 83] H. Ehrig, M. Nagl, G. Rozenberg (eds.): *Graph-Grammars and Their Application to Computer Science*, *Lect. Not. Comp. Sci.* 153, 1983
- [ENRR 87] H. Ehrig, M. Nagl, G. Rozenberg, A. Rosenfeld (eds.): *Graph-Grammars and Their Application to Computer Science*, *Lect. Not. Comp. Sci.* 291, 1987
- [EPS 73] H. Ehrig, M. Pfender, H.J. Schneider: Graph Grammars: An Algebraic Approach, *Proc. of the IEEE Conf. on Automata and Switching Theory*, Iowa City 1973, 167-180, 1973
- [EL 88] J. Engelfriet, G. Leih: Complexity of Boundary Graph Languages, Report 88-07, Leiden, 1988
- [ER 88] J. Engelfriet, G. Rozenberg: A Comparison of Boundary Graph Grammars and Context-Free Hypergraph Grammars, *Techn. Report*, Leiden, 1988
- [FKZ 76] R. Farrow, K. Kennedy, L. Zucconi: Graph Grammars and Global Program Data Flow Analysis, *Proc. 17th Ann. IEEE Symp. on Found. of Comp. Sci.*, Houston, 42-56, 1976
- [Fe 68] J. Feder: Languages of Encoded Line Patterns, *Inform. Contr.* 13, 230-244, 1968
- [Fe 71] J. Feder: Plex Languages, *Inform. Sci.* 3, 225-241, 1971
- [Fr 78] R. Franck: A Class of Linearly Parsable Graph Grammars, *Acta Informatica* 10, 175-201, 1978
- [Fu 74] K.S. Fu: *Syntactic Methods in Pattern Recognition*, Academic Press, New York, 1974
- [Ga 69] D. Gans: *Transformations and Geometries*, Appleton-Century-Crofts, New York 1969
- [Gi 85] A. Gibbons: *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1985

- [Gi 66] S. Ginsburg: The Mathematical Theory of Context-Free Languages, McGraw-Hill, New York, 1966
- [GR 62] S. Ginsburg, G. Rice: Two Families of Languages Related to ALGOL, Journ. ACM 9, 350-371, 1962
- [Go 77] J. Goldstine: A Simplified Proof of Parikh's Theorem, Discrete Mathematics 19, 235-239, 1977
- [GF 74] C. Gottlieb, A. Furtado: Data Schemata Based on Directed Graphs, Techn. Report No. 70, Dept. Comp. Sci., Univ. Toronto, 1974
- [Gr 72] S.A. Greibach: A Generalization of Parikh's Semilinear Theorem, Discrete Mathematics 2, 347-355, 1972
- [Gr 71] J. Gruska: A Characterization of Context-Free Languages, Journ. Comp. Syst. Sci. 5, 353-364, 1971
- [Ha 89] A. Habel: Graph-Theoretic Properties Compatible with Graph Derivations, Proc. Graph-Theoretic Concepts in Computer Science (WG'88), Lect. Not. Comp. Sci. 344, 11-29, 1989
- [HK 83] A. Habel, H.-J. Kreowski: On Context-Free Graph Languages Generated by Edge Replacement, Lect. Not. Comp. Sci. 153, 143-158, 1983
- [HK 85] A. Habel, H.-J. Kreowski: Characteristics of Graph Languages Generated by Edge Replacement, University of Bremen, Comp. Sci. Report No. 3/85, also in: Theor. Comp. Sci. 51, 81-115, 1987
- [HK 87a] A. Habel, H.-J. Kreowski: May We Introduce to You: Hyperedge Replacement, Lect. Not. Comp. Sci. 291, 15-26, 1987
- [HK 87b] A. Habel, H.-J. Kreowski: Some Structural Aspects of Hypergraph Languages Generated by Hyperedge Replacement, Symp. Theor. Aspects of Comp. Sci. 1987 (STACS'87), Lect. Not. Comp. Sci. 247, 207-219, 1987
- [HK 88] A. Habel, H.-J. Kreowski: Pretty Patterns Produced by Hyperedge Replacement, Proc. Graph-Theoretic Concepts in Computer Science 1987 (WG'87), Lect. Not. Comp. Sci. 314, 32-45, 1988
- [HKP 88] A. Habel, H.-J. Kreowski, D. Plump: Jungle Evaluation, Proc. Fifth Workshop on Specification of Abstract Data Types, Lect. Not. Comp. Sci. 332, 92-112, 1988
- [HKV 87] A. Habel, H.-J. Kreowski, W. Vogler: Metatheorems for Decision Problems on Hyperedge Replacement Graph Languages, to appear in Acta Informatica, short version with the title "Compatible Graph Properties are Decidable for Hyperedge Replacement Graph Languages" in: Bull. EATCS 33, 55-62, 1987
- [HKV 89] A. Habel, H.-J. Kreowski, W. Vogler: Decidable Boundedness Problems for Hyperedge-Replacement Graph Grammars, Lect. Not. Comp. Sci. 351, 275-289, 1989

- [HR 75] G.T. Herman, G. Rozenberg: *Developmental Systems and Languages*, North Holland/American Elsevier, New York, 1975
- [HU 69] J.E. Hopcroft, J.D. Ullman: *Formal Languages and Their Relation to Automata*, Addison-Wesley, Reading, Massachusetts, 1969
- [Ja 83] D. Janssens: *Node Label Controlled Graph Grammars*, Ph. D. Thesis, Univ. Antwerpen, 1983
- [JKRE 82] D. Janssens, H.-J. Kreowski, G. Rozenberg, H. Ehrig: *Concurrency of Node-Label-Controlled Graph Transformations*, University of Antwerp, Comp. Sci. Report 82-38, 1982
- [JR 80a] D. Janssens, G. Rozenberg: *On the Structure of Node-Label-Controlled Graph Languages*, *Information Sciences* 20, 191-216, 1980
- [JR 80b] D. Janssens, G. Rozenberg: *Restrictions, Extensions and Variations of NLC Grammars*, *Information Sciences* 20, 217-244, 1980
- [JR 81] D. Janssens, G. Rozenberg: *Decision Problems for Node Label Controlled Graph Grammars*, *Journ. Comp. Syst. Sci.* 22, 144-177, 1981
- [JR 82] D. Janssens, G. Rozenberg: *Graph Grammars with Neighbourhood-Controlled Embedding*, *Theor. Comp. Sci.* 21, 55-74, 1982
- [JRW 86] D. Janssens, G. Rozenberg, E. Welzl: *The Bounded Degree Problem for NLC Grammars is Decidable*, *Journ. Comp. Syst. Sci.* 33, 415-422, 1986
- [Ka 86] M. Kaul: *Syntaxanalyse von Graphen bei Präzedenz-Graph-Grammatiken*, Ph. D. Thesis, Univ. Passau, 1986
- [Kl 56] S.C. Kleene: *Representation of Events in Nerve Nets and Finite Automata*. In: C. Shannon, J. McCarthy (eds.): *Automata Studies*, *Annales of Math. Studies* 34, Princeton, N.J., 3-40, 1956
- [Kr 77a] H.-J. Kreowski: *Manipulationen von Graphmanipulationen*, Ph. D. Thesis, Techn. Univ. Berlin, Comp. Sci. Dept., 1977
- [Kr 77b] H.-J. Kreowski: *Transformations of Derivation Sequences in Graph Grammars*. *Lect. Not. Comp. Sci.* 56, 275-286, 1977
- [Kr 79] H.-J. Kreowski: *A Pumping Lemma for Context-Free Graph Languages*, *Lect. Not. Comp. Sci.* 73, 270-283, 1979
- [Kr 86] H.-J. Kreowski: *Rule Trees Represent Derivations in Edge Replacement Systems*, in G. Rozenberg, A. Salomaa (eds.): *The Book of L*, Springer-Verlag, Berlin, 217-232, 1986
- [Kr 87a] H.-J. Kreowski: *Rule Trees Can Help to Escape Hard Graph Problems*, Technical Report, Univ. Bremen, 1987
- [Kr 87b] H.-J. Kreowski: *Is Parallelism Already Concurrency? Part 1: Derivations in Graph Grammars*, *Lect. Not. Comp. Sci.* 291, 343-360, 1987

- [KR 88] H.-J. Kreowski, G. Rozenberg: On Structured Graph Grammars, Techn. Report , Univ. Bremen, 1988
- [La 88a] C. Lautemann: Decomposition Trees: Structured Graph Representation and Efficient Algorithms, Proc. CAAP'88, Lect. Not. Comp. Sci. 299, 28-39, 1988
- [La 88b] C. Lautemann: Efficient Algorithms on Context-Free Graph Languages, Proc. ICALP'88, Lect. Not. Comp. Sci. 317, 362-378, 1988
- [Le 85] T. Lengauer: Efficient Solution of Biconnectivity Problems on Hierarchically Defined Graphs, Proc. WG'85, Trauner Verlag, Linz, 201-215, 1985
- [LW 88] T. Lengauer, E. Wanke: Efficient Analysis of Graph Properties on Context-Free Graph Languages, Proc. ICALP'88, Lect. Not. Comp. Sci. 317, 379-393, 1988
- [Li 85] U. Lichtblau: Decompilation of Control Structures by Means of Graph Transformations, Lect. Not. Comp. Sci. 185, 284-297, 1985
- [MR 87a] M.G Main, G. Rozenberg: Handle NLC Grammars and R.E. Languages, Journ. Comp. Syst. Sci. 35, 195-205, 1987
- [MR 87b] M.G Main, G. Rozenberg: Fundamentals of Edge-Label Controlled Graph Grammars, Lect. Not. Comp. Sci. 291, 411-426, 1987
- [Ma 77] B.B. Mandelbrot: Fractals, Freeman, San Francisco, 1977
- [Mo 70] U. Montanari: Separable Graphs, Planar Graphs and Web Grammars, Inf. Contr. 16, 243-267, 1970
- [MR 87] U. Montanari, F. Rossi: An Efficient Algorithm for the Solution of Hierarchical Networks of Constraints, Lect. Not. Comp. Sci. 291, 440-457, 1987
- [Na 79] M. Nagl: A Tutorial and Bibliographical Survey on Graph Grammars, Lect. Not. Comp. Sci. 73, 70-126, 1979
- [Pa 61] R.J. Parikh: Language Generating Devices, Quarterly Progress Report, No. 60, Research Laboratory of Electronics, M.I.T., 199-212, 1961
- [Pa 66] R.J. Parikh: On Context-Free Languages, Journ. ACM 13, 570-581, 1966
- [Pa 72] T. Pavlidis: Linear and Context-Free Graph Grammars, Journ. ACM 19, 1, 11-23, 1972
- [PR 69] J.L. Pfaltz, A. Rosenfeld: Web Grammars, Proc. Int. Joint Conf. Art. Intelligence, 609-619, 1969
- [Pr 71] T.W. Pratt: Pair Grammars, Graph Languages and String- to-Graph Translations, Journ. Comp. Syst. Sci. 5, 560-595, 1971
- [Pr 86] P. Prusinkiewicz: Graphical Applications of L-Systems, Proc. of Graphics Interface'86 - Vision Interface'86, 247-253, 1986

- [RS 86] N. Robertson, P.D. Seymour: Graph Minors II, Algorithmic Aspects of Tree Width, *Journal of Algorithms* 7, 309-322, 1986
- [Ro 74] D.J. Rose: On Simple Characterizations of k -Trees, *Discrete Math.* 7, 317-322, 1974
- [Ro 75] B.K. Rosen: Deriving Graphs from Graphs by Applying a Production, *Acta Informatica* 4, 337-357, 1975
- [RM 72] A. Rosenfeld, D. Milgram: Web Automata and Web Grammars, *Machine Intelligence* 7, 307-324, 1972
- [RW 86a] G. Rozenberg, E. Welzl: Boundary NLC Graph Grammars — Basic Definitions, Normal Forms, and Complexity, *Inf. Contr.* 69, 136-167, 1986
- [RW 86b] G. Rozenberg, E. Welzl: Graph Theoretic Closure Properties of the Family of Boundary NLC Graph Languages, *Acta Informatica* 23, 289-309, 1986
- [Sa 73] A. Salomaa: *Formal Languages*, Academic Press, New York, 1973
- [Sh 69] A.C. Shaw: A Formal Description Schema as a Basis for Picture Processing Systems, *Inf. Contr.* 14, 9-52, 1969
- [Sl 82] A.O. Slisenko: Context-Free Graph Grammars as a Tool for Describing Polynomial-time Subclasses of Hard Problems, *Information Processing Letters* 14, 52-56, 1982
- [SM 80] I. Suzuki, T. Murata: Stepwise Refinements of Transitions and Places, *Informatik-Fachberichte* 52, Springer, 136-141, 1980
- [Sy 79] M. Syslo: Characterization of Outerplanar Graphs, *Discrete Math.* 26, 47-53, 1979
- [Tu 83] G. Turán: On the Complexity of Graph Grammars, *Acta Cybern.* 6, 271-281, 1983
- [Ue 78] T. Uesu: A System of Graph Grammars which Generates all Recursively Enumerable Sets of Labelled Graphs, *Tsukuba J. Math.* 2, 11-26, 1978
- [Va 79] R. Valette: Analysis of Petri Nets by Stepwise Refinements, *Journ. Comp. Syst. Sci.* 18, 35-46, 1979
- [Vo 88] W. Vogler: A Note on Hyperedge Replacement and BNLC Graph Grammars, Manuscript, 1988
- [Yn 71] M.K. Yntema: Cap Expressions for Context-Free Languages, *Inf. Contr.* 18, 311-318, 1971

Index

algebraic approach, 42, 169

Arbib, M. A., 173

application condition, 177

– left-sided, 177

– right-sided, 177

A-substitution, 66

– iterated, 66

attaching point, 36

axiom, 17

base, 47, 187

Bauderon, M., 41

Berlin approach, 42, 169

boundary NLC grammar, 43

BNLC grammar, 43

Boolean operation, 142

boundary NLC grammar, 43

boundedness problems, 165

bundle of string graphs, 121

Bunke, H., 180

canonical

– derivation, 57

– dissection, 56

cellular graph grammar, 42

Characterization Theorem, 66

chemical structures, 20

closed

– under Boolean operations, 142

– under isomorphisms, 11, 141, 161

color, 6

coloring, 139

compatible

– function, 161

– predicate, 141

composition, 85

– iterated, 85

compressed bundle, 121

connected, 133

connectivity, 100

Connectivity Theorem, 100, 101

contact condition, 169

context condition, 178

– concrete, 179

– negative, 178

– positive, 178

context enlargement, 179

context-free

– cellular graph grammar, 42

– graph grammar, 37ff

– plex grammar, 37

– string grammar, 35

– string-graph grammar, 119

Context-Freeness Lemma, 49, 52, 130

context-sensitive, 173–175

control-flow graph, 7, 18

Courcelle, B., 41, 153

cut of a derivation tree, 56

cycle, 156

– Eulerian, 137

cycle-free grammar, 83

decidable, 147

decision problems, 147, 151

definable graph property, 153

degree, 102, 158

Della Vigna, P., 38

density of a hypergraph, 155

derivation, 16, 170

– canonical, 56

– direct, 16, 170

– dummy, 47

– non-trivial, 83

– parallel, 47

– sequential, 16

– valid, 47

derivation tree, 54

– cut of, 56

– dissection of, 56

– result of, 55

direct derivation, 16

dissection of a derivation tree, 56

dummy derivation, 16

edge, 6, 9

edge coloring, 139

- edge-label controlled, 45
- edge replacement, 4, 45
- edge-replacement grammar, 45, 108
- edge-replacement language, 108
- Ehrig, H., 37, 42, 169
- ELC grammar, 45
- embedding of derivations, 50
- embedding condition, 178
- empty production, 77
- Emptiness Problem, 77
- Engelfriet, J., 43
- equation, 67
- equivalent grammar, 17
- ETOL system, 181, 190
- Eulerian, 118, 135
 - cycle, 137
 - path, 137
- external node, 8

- Farrow, R., 18
- Feder, J., 20, 36
- fibre of a derivation, 49, 131
- figure, 184
- figure-generating grammar, 188
- figure language, 188
- finite graph property, 153
- Finiteness Problem, 92
- fixed point, 69
- Fixed-Point Theorem, 70
- forbidden structures, 178
- frame, 10
- functional expression, 7

- Gans, D., 187
- generative power, 175, 180, 183
- Ginsburg, S., 69
- Ghezzi, C., 38
- gluing,
 - condition, 178
 - diagram, 173
 - hypergraph, 169
- grammar, 17
 - cycle-free, 83
 - figure-generating, 188
 - parallel, 181
 - proper, 83
 - reduced, 73
 - typed, 27
 - usual, 25
 - well-formed, 31
 - with application conditions, 177
 - without empty productions, 77
 - without single productions, 81
- graph, 9, 117, 171
 - bipartite, 22, 118
 - complete, 93, 117, 171
 - connected, 133, 171
 - Eulerian, 118, 135, 171
 - Hamiltonian, 118, 172
 - maximal outerplanar, 110
 - multi-pointed, 9
 - planar, 118
 - semi-connected, 133
 - simple, 100
- graph grammar
 - algebraic approach, 42
 - cellular, 42
 - context-free, 37-42
 - edge-label controlled, 45
 - node-label controlled, 43
 - edge replacement, 108
 - node replacement, 38-39
- graph language, 108
- Graph Problem, 110
- graph property
 - compatible, 141
 - definable, 153
 - finite, 153
- grid, 116
- Gruska, J., 66

- handle, 10
- handle NLC grammar, 44
- Herman, G. T., 181
- hierarchical networks, 41
- Hierarchy Theorem, 116, 126
- Hilbert patterns, 182
- homogeneous language, 17, 61

- homomorphism, 60
- HRG/HRL 17
- hyperedge, 5–6, 184
 - deformed, 31
 - well-formed, 30
- hyperedge replacement, 12, 186
- hyperedge-replacement grammar, 17
 - cycle-free, 83
 - figure-generating, 188
 - parallel, 181
 - proper, 83
 - reduced, 73
 - typed, 27
 - usual, 25
 - well-formed, 31
 - without empty productions, 77
 - without single productions, 81
- hyperedge-replacement language, 17
- hypergraph, 6
 - multi-pointed, 8
 - simple, 100
 - well-formed, 31
 - well-proportioned, 121
- hypergraph grammar, 170
 - context-free, 17, 174
 - context-sensitive, 174
 - monotone, 174
 - underlying, 177
 - with application conditions, 177
- hypergraph morphism, 10, 177
- hypergraph production, 169
 - context-free, 173
 - context-sensitive, 173
 - monotone, 174
- hypergraph language, 11
 - closed under isomorphisms, 11
 - context-free, 174
 - context-sensitive, 174
 - finite, 11
 - generated, 17, 170, 177
 - homogeneous, 17
 - monotone, 174
- hypergraph-replacement grammar, 169
 - with application conditions, 177
- identification condition, 169
- independence number, 166
- inscribed square, 189
- internal node, 8
- intersection problems, 149
- inverse production, 170
- isomorphic, 10
- isomorphism, 10
- iterated
 - substitution, 61, 66
 - composition, 85
- Janssens, D., 43, 152
- joint embedding, 50, 131
- Joint-Embedding Lemma, 50
- Kennedy, K., 18
- Kleene, S. C., 66
- Kleene-Type Characterization, 66
- Kreowski, H.-J., 51, 57, 153
- k -bounded, 138
- k -edge-coloring, 139
- k -tree, 113
- label, 6
- labeling, 6
- language, 11
 - finite, 11
 - homogeneous, 17, 61
- language family, 70
- Lautemann, C., 44, 153
- least fixed point, 69
- left-hand side, 16, 169
- left-occurrence map, 169
- left-sided application condition, 177
- Leih, G., 43
- Lengauer, T., 42, 153
- linear set, 96
- linear-time algorithm, 146
- Main, M. G., 45, 46
- Mandelbrot, B. B., 191
- Manes, E. G., 173

- Membership Problem, 81
- metatheorem
 - for boundedness problems, 165
 - for decision problems, 147, 151
 - for intersection problems, 149
- (m, n) -edge, 6
- (m, n) -graph, 9
- (m, n) -hypergraph, 8
- (m, n) -handle, 10
 - induced 10
- (m, n) -hypergraph 8
- Milgram, D., 180
- monotone grammar, 174
- Montanari, U., 41, 180
- morphism, 10, 177
- m th order,
 - context-free graph grammar, 37
 - structure, 37
- Murata, T., 14
- n attaching-point entity, 36
- NAPE, 36
- neighborhood uniform, 43
- NLC grammar, 43–44
- NCE grammar, 43
 - boundery, 43
 - context-free, 43
 - Church–Rosser, 43
 - handle, 44
 - neighborhood uniform, 43
- node, 6
 - external, 8
 - internal, 8
- node-label controlled, 43–44
- node replacement, 38–41, 43–44
- non-compatible, 151
- nonterminal, 16
- non-trivial automorphism group, 152
- Normal-Form Theorem, 121
- NUNLC grammar, 43
- occurrence, 170
- occurrence map, 169
- order of a grammar/language, 86
- parallel derivation, 47
- parallel grammar, 181
- Parikh, R. J., 95
- Parikh mapping, 95
 - modified, 99
- Parikh's Theorem, 96
- path, 156
 - Eulerian, 137
- Pavlidis, T., 37
- Petri net, 7, 14
- plex grammar, 36
- plex structure, 36
- Post Correspondence Problem, 152
- Pratt, T. W., 38
- production, 16, 169, 187
 - inverse, 170
- production base, 47
- proper grammar, 83
- property
 - compatible, 141
 - definable, 153
 - finite, 153
- pullback, 173
- Pumping Lemma, 87
- pushout, 173
- quotient hypergraph, 32
- reduced grammar, 73
- Reducedness Theorem, 73
- replacement
 - of hyperedges, 12, 186
 - of hypergraphs, 169
- restriction of derivations, 49
- Restriction Lemma, 49
- right-hand side, 16, 169
- right-occurrence map, 170
- right-sided application condition, 177
- Rice, G., 69
- Rose, D. J., 113
- Rosenfeld, A., 180
- Rossi, F., 41
- Rozenberg, G., 43ff, 152, 181

- semi-connected, 133
- semilinear set, 96
- semi-structured programs, 18
- sequential derivation, 16
- sequentialization, 48
- Sequentialization Theorem, 48
- simple hypergraph, 100
- single production, 81
- singleton, 10
- Sierpinski triangle, 190
- size of a hypergraph, 87
- Slisenko, A. O., 39, 153
- snail, 106, 118, 173
- snowflake, 182
- solution of an equation system, 67
- source, 6
- square grid, 94, 118, 173
- subhypergraph, 10
- star, 34
- star-replacement grammar, 34
- string grammar, 35, 119
- string graphs, 9, 23, 108
 - bundle of, 121
 - compressed bundle of, 121
 - induced, 9
- string-graph grammar, 119
- string-graph language, 119
- String-Graph Problem, 119
- sublinear growing, 93
- substitution, 59, 66
- iterated, 61, 66
- Suzuki, I., 14
- system of equations, 67
 - associated, 70
- table, 181
- target, 6
- tentacle of a hyperedge, 5
- terminal, 17
- transformation
 - affine, 187
 - geometrical, 187
 - identical, 187

- similarity, 187
- topological, 187
- topological equivalence, 186
- totally disconnected, 132
- type
 - of a hyperedge, 6
 - of a hypergraph, 8
 - of a production, 16
- typed
 - grammar, 27
 - production, 54
- Type-Modification Theorem, 27
- Typification Theorem, 27

- Uesu, T., 175, 180
- unsolvable problems, 152
- usual grammar, 25

- Valette, R., 14
- valid derivation, 47
- vertex, 6
- Vogler, W., 44

- Wanke, E., 42, 153
- Wankmüller, F., 178
- well-formed
 - grammar, 31
 - hyperedge, 30
 - hypergraph, 31
- Well-Formedness Theorem, 31
- well-proportioned hypergraph, 121
- Welzl, E., 42, 153
- wheel, 105

- X -homomorphism, 60
- X -substitution, 59
 - iterated, 61
- $X(m, n)$ -handle, 85
- $X(m, n)$ -handled, 85

- Yntema, M. K., 66

- Zucconi, L., 18

Annegret Habel

Curriculum Vitae

Born:	Jan. 1955, Jork (Kreis Stade)
Maiden name:	Annegret Behr
Married:	Dec. 1976, to Christopher Habel

Education:

Oct. 1973 – Oct. 1976:	University of Göttingen fields: Mathematics and Physics
Oct. 1976 – Sept. 1981:	Technical University of Berlin, fields: Mathematics and Computer Science
Oct. 1981:	Graduation in Mathematics (Dipl. Math.) thesis topic: "Komposition und Dekomposition von Produktionen zur Manipulation von Graphen" (supervised by Prof. Dr. Harmut Ehrig)
April 1989:	Ph. D. in Computer Science (Dr. rer. nat.) thesis topic: Hyperedge Replacement: Grammars and Languages (supervised by Prof. Dr. Hans-Jörg Kreowski)

Professional Experience:

Oct. 1981 – Sept. 1986:	Assistant Department of Computer Science Technical University of Berlin
Oct. 1986 – present:	Assistant Department of Mathematics and Computer Science University of Bremen

